

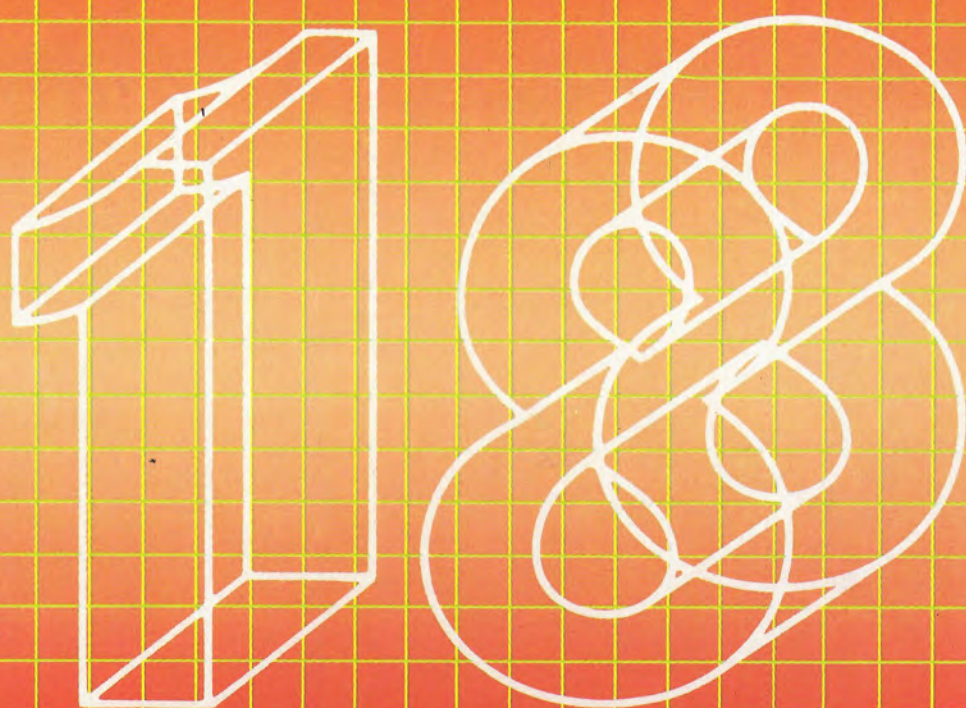
BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,

IBM, SPECTRUM, COMMODORE Y MSX



APLICACIONES PRACTICAS EN BASIC.

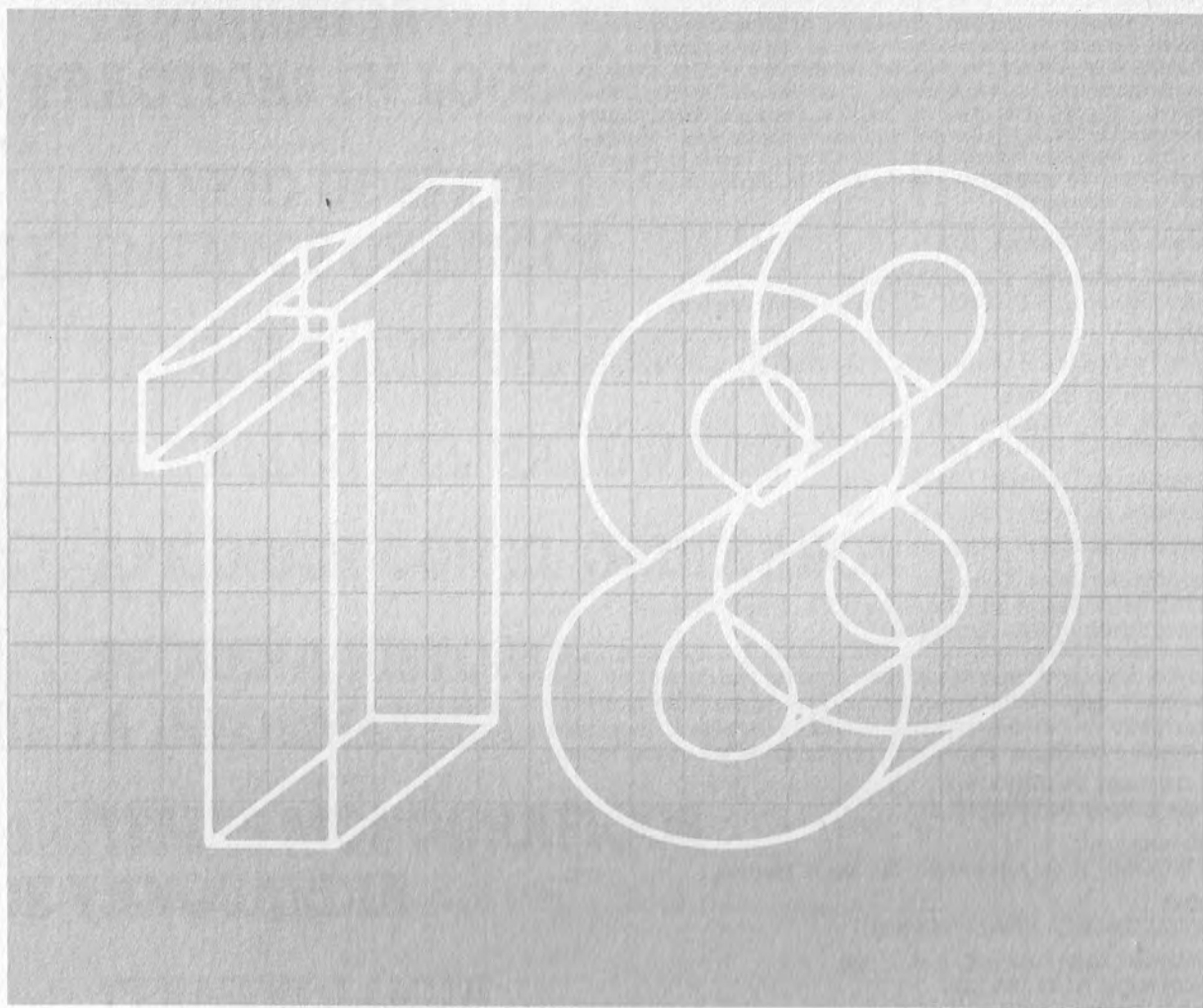
PROCEDIMIENTOS Y RUTINAS

EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 18. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-110-X

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

MATEU CROMO. PINTO (Madrid).

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Mayo, 1987

P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	15
■ TRUCOS Y RUTINAS BASICAS	23
■ EL TALLER DEL HARDWARE	33
■ APRENDER CON EL ORDENADOR	41
■ PEQUEÑA HISTORIA DE LA INFORMATICA	51
■ TEMAS MONOGRAFICOS DE VANGUARDIA	55
■ TERMINOLOGIA	58
■ VOCABULARIO DE INFORMATICA	59

EXPERIENCIA Y PRACTICAS EN LOGO

Condicionales



A conocer la forma de establecer, dentro de un proceso, una condición, para que, dependiendo de su resultado, se realice una acción u otra.

Conozcamos ahora cómo podemos establecer más de una condición dentro del mismo proceso.

Definimos los procedimientos necesarios para obtener los catorce resultados de una quiniela de fútbol. La forma de hacerlo es la siguiente:

A una variable le asignamos un valor elegido al azar comprendido entre 0 y 2. Dependiendo del valor que contenga la variable, la Tortuga deberá dibujar en la pantalla:

- Un 1, si la variable contiene el valor uno.
- Un 2, si la variable contiene el valor dos.
- Una X, si la variable contiene el valor cero.

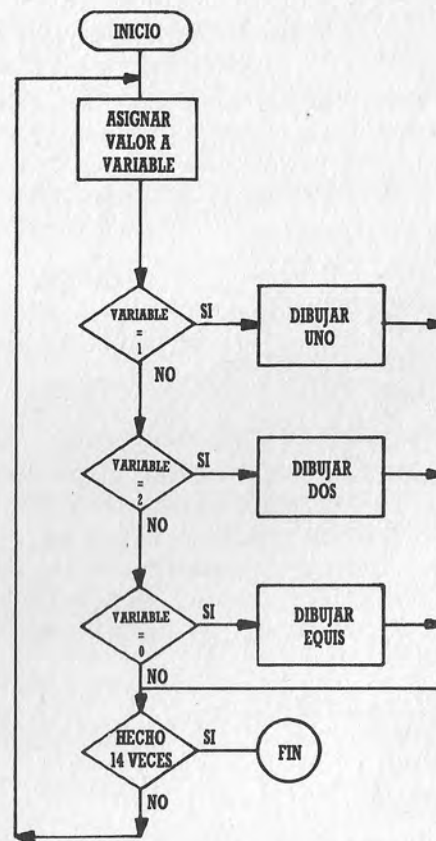
Analizamos el proceso y vemos las diferentes partes de que consta:

- 1.º Asignar un valor al azar a una variable.
- 2.º Ver si el número es UNO. Si lo es, dibujar un uno.
- 3.º Ver si el número es DOS. Si lo es, dibujar un dos.

4.º Ver si el número es CERO. Si lo es, dibujar una equis.

Observa que tenemos tres condiciones y cada una de ellas tiene que realizar una acción diferente en el caso que se cumpla.

Hacemos la representación gráfica del proceso mediante un organigrama:



Una condición puede expresarse mediante operadores lógicos o de relación.

EXPERIENCIA Y PRACTICAS EN LOGO

Definimos ahora los procedimientos necesarios para realizarlo:

En primer lugar, definimos los que van a dibujar el UNO, el DOS y la EQUIS:

? PARA UNO

- > AV 15
- > GI 135
- > AV 5
- > FIN

? PARA DOS

- > GI 90 AV 5
- > GD 90 AV 7
- > GD 90 AV 5
- > GI 90 AV 7
- > GI 90 AV 5
- > FIN

? PARA EQUIS

- > PONPOS [-7 15]
- > SL
- > PONY O
- > BL
- > PONPOS [0 15]
- > FIN

A continuación definimos el procedimiento con el que vamos a obtener un número aleatorio y dependiendo del que sea, dibujar en la pantalla el signo correspondiente:

? PARA SIGNOS

- > HAZ "V AZAR 3
- > SI :V = 1 [UNO]
- > SI :V = 2 [DOS]
- > SI :V = 0 [EQUIS]
- > FIN

En este procedimiento es donde se establecen las tres condiciones. Cada una de ellas se debe expresar mediante un SI seguido de la condición que se precisa.

Por último, definimos un procedimiento para que el proceso que se realiza en SIGNOS se repita catorce veces y así poder obtener los catorce resultados.

? PARA QUINIELA

- > REPITE 14 [SL BP BL SIGNOS ESPERA 50]
- > FIN

Para ejecutarlo bastaría con introducir:

? QUINIELA

El procedimiento SIGNOS puede ser reducido. La última pregunta no sería necesaria hacerla, ya que si el valor de la variable no es ni uno ni dos, a la fuerza tiene que ser cero:

? PARA SIGNOS

- > HAZ "V AZAR 3
- > SI :V = 1 [UNO]
- > SI :V = 2 [DOS]
- > EQUIS
- > FIN

Veamos qué ocurre en este caso:

Supongamos que la variable contiene el valor uno. El resultado al primer SI sería CIERTO y se realizaría la acción (dibujar un uno). A continuación pasa a la línea siguiente y como la variable no es igual a dos, no se realizaría la acción, pasando a la línea siguiente. Como ésta sólo contiene una llamada al procedimiento EQUIS, se dibujaría una equis encima del uno, lo cual no tendría que ocurrir.

Si la variable contiene el valor dos, se dibujaría un dos y a continuación también una equis, lo que tampoco tendría que ocurrir.

Para evitar esto, se tendría que conseguir que no se continuase con la ejecución del procedimiento una vez se haya dibujado un uno o un dos.

Esto lo podemos lograr gracias a la orden ALTO.

La orden ALTO detiene el procedimiento que se está ejecutando y devuelve el control al procedimiento que lo ha llamado, si existe. De no ser así, la ejecución se detiene por completo.

En el procedimiento SIGNOS lo único que hay que hacer es devolver el control al procedimiento QUINIELA una vez que se haya dibujado el uno o el dos, para así no dibujar la equis:

? PARA SIGNOS

- > HAZ "V AZAR 3
- > SI :V = 1 [UNO ALTO]
- > SI :V = 2 [DOS ALTO]
- > EQUIS
- > FIN

El procedimiento SIGNOS puede acortarse aún más.

? PARA SIGNOS

Una condición puede formar parte de una determinada acción.

```
> HAZ "V AZAR 3
> SI :V = 1 [UNO] [SI :V = 2 [DOS]
[EQUIS]]
> FIN
```

Se establece, en primer lugar, una condición. Si ésta se cumple, la acción a tomar es dibujar un UNO. Si no se cumple, realiza la acción que se encuentra en los segundos corchetes. Esta acción contiene a su vez otra condición (:V = 2) y dos posibles acciones a hacer dependiendo del resultado. Si se cumple, se dibuja un dos, y si no, se dibuja una equis.

Observa cómo las condiciones pueden formar parte de las acciones.

■ Entrada de caracteres

Todos los lenguajes de programación disponen de órdenes o instrucciones que permiten leer datos de los periféricos, como pueden ser: el teclado, un fichero del disco...

En LOGO la orden LEECAR lee un carácter en el archivo que en ese momento esté activado como archivo de lectura, por defecto el carácter es leído del teclado. Y mientras no pulsemos una tecla, LEECAR estará esperándola, lo cual implica que la ejecución del procedimiento no avanzará, a no ser que combiemos esta orden con TECLA?, pero de esto ya hablaremos más adelante.

```
? LEECAR
```

El ordenador se quedará esperando para leer un carácter, en nuestro caso desde el teclado. Cuando una tecla haya sido pulsada, la espera finalizará.

Pero algo no funciona bien, el LOGO nos ha devuelto el mensaje:

```
NO SE QUE DEBO HACER CON K
(suponemos que se ha pulsado la letra
K)
```

Lo que ha ocurrido es que no hemos precisado qué queríamos hacer con el carácter leído.

Podemos escribirlo:

```
? ESCRIBE LEECAR
```

Esta orden le indica al ordenador que

debe esperar a que un carácter sea pulsado, y luego que lo escriba.

Podemos guardarlo en una variable:

```
? HAZ "S LEECAR
```

Cuando hayamos pulsado una tecla, su valor se almacenará en la variable S.

Utilizarlo de operando:

```
? ESCRIBE 5 + LEECAR
```

Si pulsamos el 2, en la pantalla aparece un 7.

Utilizarlo dentro de una expresión lógica.

```
? SI LEECAR = "S [BP]
```

Si pulsamos una S la pantalla se borrará.

Con estos ejemplos habrás comprobado que mientras el ordenador está esperando que se pulse una tecla, no aparece ningún aviso en la pantalla; lo único que vemos es el cursor.

A continuación borramos la pantalla de texto y le decimos al ordenador que espere la entrada de un carácter.

```
? PT
```

```
? BT HAZ "D LEECAR
```

mientras no pulsemos una tecla en la pantalla, sólo veremos el cursor.

NOTA:

En el LOGO del SPECTRUM la sintaxis de esta primitiva es

LEECARACTER o LCAR

Antes hemos mencionado el nombre de otra primitiva, una primitiva que guarda relación con LEECAR, nos referimos a:

```
TECLA?
```

Si piensas que devuelve un valor lógico, has acertado.

Devuelve CIERTO si una tecla ha sido pulsada y un carácter espera ser leído por la orden LEECAR; si no, devuelve FALSO.

```
? ESCRIBE TECLA?
```

```
FALSO
```

La posibilidad de introducir datos a un procedimiento con la primitiva LEECAR resuelve muchos problemas.

EXPERIENCIA Y PRACTICAS EN LOGO

? ESPERA 40

? ESCRIBE TECLA?

Si mientras se ejecuta ESPERA 40 pulsamos una tecla, la orden ESCRIBE TECLA? devolverá CIERTO; si no, devolverá FALSO.

? MT ESPERA 50 SI TECLA? [MT] [OT]

Le hemos dicho al ordenador que espere un cierto tiempo; a continuación que mire si se ha pulsado alguna tecla; si así es, aparece la Tortuga; si no, desaparece.

Una vez tecleada la línea anterior y hasta que la ejecución de la orden ESPERA 50 haya terminado, tenemos tiempo para pulsar una tecla; si no lo hacemos, la Tortuga se ocultará.

Prueba todo lo que hemos visto con estos ejemplos.

— Procedimiento P1

? PARA P1

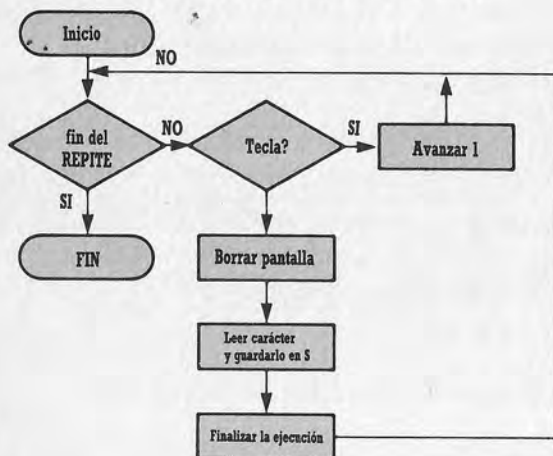
> REPITE 100 [SI TECLA? [BP HAZ "S LEECAR ESCRIBE :S] [AV 1]]

> FIN

El procedimiento P1 repite 100 veces la pregunta ¿se ha pulsado alguna tecla?; si la contestación es afirmativa (CIERTO), borramos la pantalla, asignamos a la variable S el valor de la tecla que hemos pulsado y escribimos dicho valor en la pantalla; si la contestación, por el contrario, es negativa (FALSO), la Tortuga avanza un paso.

Si ejecutamos este procedimiento y no pulsamos ninguna tecla, la Tortuga avanzará 100 pasos sin que se borre la pantalla y S no tendrá ningún valor.

Observa el organigrama de P1:



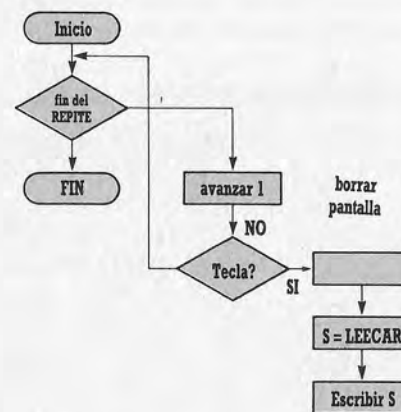
— Procedimiento P2

? PARA P2

> REPITE 100 [AV 1 SI TECLA? [BP HAZ "S LEECAR ALTO]]

> FIN

El procedimiento P2 es similar al anterior; la principal diferencia es que cuando pulsamos una tecla abandonaremos la ejecución del procedimiento.



— Procedimiento P3

? PARA P3

> HAZ "S "

> ESPERA 40

> SI TECLA? [BP HAZ "S LEECAR]

> SI :S = "*" ALTO

> SI Y (:S > 2) (:S < 6) [POLI :S] [ESCRIBE "INCORRECTO]

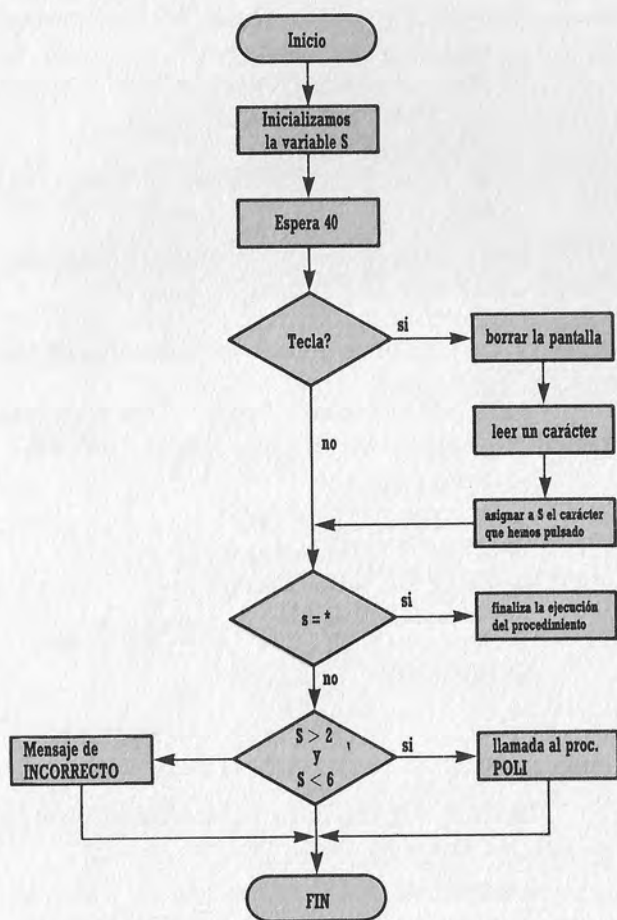
> FIN

Con el procedimiento (P3) podemos elegir desde el teclado el número de lados que queremos que tenga el polígono regular que el procedimiento POLI dibuja en la pantalla.

El polígono regular sólo se dibujará en el caso que pulsemos una tecla y que ésta corresponda a un número mayor que dos y menor que 7; esto significa que sólo podemos dibujar polígonos de tres a seis lados, ambos inclusive.

Lo explicamos con un organigrama:

Podemos detener la ejecución de un procedimiento en el momento que deseemos.



El procedimiento que llamamos desde P3 dibuja un polígono regular cuyo número de lados determinamos nosotros desde el teclado.

```

? PARAR POLI :N
> REPITE :N [AV 30 GD 360 / :N]
> FIN
  
```

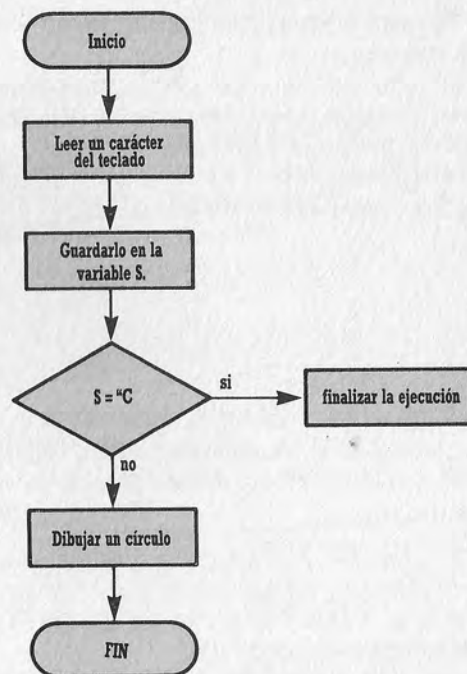
—Procedimiento P4

Y por último, el procedimiento P4, que lee un carácter del teclado, y si el carácter pulsado es la letra C, el procedimiento finaliza; si no, se dibuja un círculo.

? PARA P4

```

> HAZ "S LEECAR
> SI :S = "C [ALTO]
> REPITE 36 [AV 1 GD 10]
> FIN
  
```



Cuadro resumen

— ALTO

Esta primitiva detiene la ejecución del procedimiento que la contiene, devolviendo el control al procedimiento desde el que ha sido llamado.

— LEECAR

Lee un carácter del archivo que en ese momento esté activado como archivo de lectura; por defecto el carácter se leerá del teclado.

Al encontrar LEECAR, la ejecución de un procedimiento se detiene, hasta que el carácter no se introduzca.

— TECLA?

Devuelve CIERTO si un carácter espera ser leído; si no, devuelve FALSO.

Ejercicios

1. Define dos procedimientos, uno que devuelva el mayor de dos números y otro que devuelva el mayor de tres números.

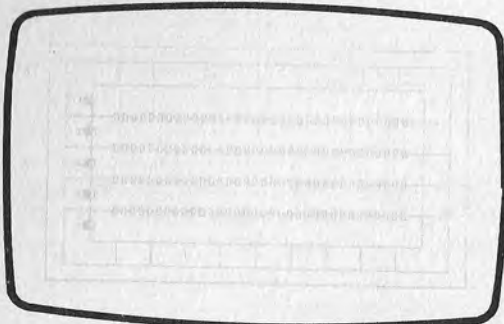
Una condición puede estar formada por expresiones lógicas y de relación.

EXPERIENCIA Y PRACTICAS EN LOGO

Ten en cuenta que los números pueden ser cualesquiera.

2. Sin utilizar los procedimientos con entradas, intenta conseguir los resultados de cualquiera de las tablas de multiplicar.

3. Intenta hacer el siguiente dibujo.



4. Define los procedimientos necesarios para que, dependiendo de los valores asignados al azar a dos variables, se dibuje la ficha de dominó correspondiente.

5. ¿Son correctas las órdenes siguientes?

- ? HAZ "A 1
- ? HAZ "B 0
- ? SI :A = 1 [SI :B > 10 [HAZ "A :B]]
- ESCRIBE LEECAR = :A
- ? PARA A
- > AV 10 GD 90
- > REPITE 3 [AV 10 GD 120]
- > ALTO
- > FIN
- ? HAZ "V AZAR 10
- ? SI :V > 2 ALTO [RE :V]
- ? HAZ "A LEECAR
- ? ESPERA 25
- ? SI TECLA [ESCRIBE :A + LEECAR]
- [ESCRIBE :A]
- ? HAZ "A 1
- ? SI TECLA? = :A [ESCRIBE "CORRECTO]

Solución a los ejercicios

1:

El procedimiento ORD devuelve el mayor de dos números cualesquiera.

? PARA ORD

- > ESCRIBE "PRIMERO?
- > HAZ "A LEECAR
- > ESCRIBE "SEGUNDO?
- > HAZ "B LEECAR
- > SI :A > :B [ESCRIBE :A] [ESCRIBE :B]
- > FIN

Los números los introducimos en variables, a través de la primitiva LEECAR.

Para guiarnos un poco a la hora de introducir los números hemos incluido dos órdenes ESCRIBE.

Para seleccionar el mayor de tres números ampliamos el procedimiento anterior.

? PARA ORD3

- > ESCRIBE "PRIMERO?
- > HAZ "A LEECAR
- > ESCRIBE "SEGUNDO?
- > HAZ "B LEECAR
- > SI :A > :B [HAZ "F :A] [HAZ "F :B]
- > ESCRIBE "TERCERO?
- > HAZ "C LEECAR
- > SI :C > :F [ESCRIBE :C] [ESCRIBE :F]
- > FIN

Si después de haber introducido en el ordenador estos procedimientos tecleas:

? ORD3

Aparecerá en pantalla:

PRIMERO?

nos indica que introduzcamos el primer número.

Esto mismo ocurre con SEGUNDO? y TERCERO?

2:

El siguiente procedimiento muestra los resultados de la tabla de multiplicar del número que nosotros pulsemos.

? PARA TABLA

- > ESCRIBE "TABLA
- > HAZ "A LEECAR
- > HAZ "C 0
- > REPITE 10 [HAZ "C :C + 1 HAZ "R :A
- *:C ESCRIBE :R]
- > FIN

Ejecutemos el procedimiento TABLA.

? TABLA

Con la orden SI podemos controlar nuestros procedimientos.

observa que la ejecución del procedimiento se ha detenido, es el momento de introducir el número, por ejemplo, el 6.

(pulsa 6)

han aparecido los números del 1 al 10 multiplicados por 6.

Ampliamos la acción del procedimiento anterior con el procedimiento TABLAS.

Su función es la siguiente:

1. Preguntar cuántas tablas queremos obtener.

2. Ejecutar TABLA tantas veces como hayamos indicado.

? PARA TABLAS

> PT

> BT

> HAZ "A LEECAR

> ESCRIBE :A

> REPITE :A [TABLA]

> FIN

3:

Vamos a ir dibujando la piscina definiendo procedimientos que dibujan distintas partes de ella.

En primer lugar, vamos a dibujar el contorno de la piscina. Para ello definimos el procedimiento DIMEN, el cual hace tres llamadas al procedimiento RECTAN después de colocar a la Tortuga en los lugares necesarios:

? PARA DIMEN

> SL

> PONPOS [-120 -60]

> RECTAN 150 240

> SL

> PONPOS [-110 -50]

> RECTAN 130 220

> SL

> PONPOS [-95 -35]

> RECTAN 100 190

> GD 90 AV 15 GI 90

> REPITE 2 [REPITE 9 [BL RE 15 AV 15 GD 90 SL AV 20 GI 90] SL PONPOS [-80 80] PONRUMBO 0]

> SL

> PONPOS [-95 -15]

> PONRUMBO 90

> REPITE 2 [REPITE 4 [BL RE 15 AV 15

SL GI 90 AV 20 GD 90] SL PONPOS [110 -15]]

> FIN

El procedimiento RECTAN dibuja rectángulos de diferentes tamaños. Utiliza dos variables de entrada cuyos valores son dados en el procedimiento anterior:

? PARA RECTAN :A :L

> BL

> REPITE 2 [AV :A GD 90 AV :L GD 90]

> FIN

La variable :A determina la anchura del rectángulo y la variable :L la longitud.

A continuación definimos los procedimientos que van a dibujar las corcheras. Con CORCHERAS vamos situando a la Tortuga en los lugares donde debe empezar a dibujarlas y luego se realiza una llamada a CORCHO:

? PARA CORCHERAS

> SL

> PONPOS [-95 -15]

> CORCHO

> SL

> PONPOS [-95 5]

> CORCHO

> SL

> PONPOS [-95 25]

> CORCHO

> SL

> PONPOS [-95 45]

> CORCHO

> SL

> FIN

El procedimiento corcho es el que dibuja las corcheras:

? PARA CORCHO

> PONRUMBO 90

> BL

> AV 10

> REPITE 34 [GD 90 AV 2 GI 90 REPITE 2 [AV 2 GI 90 AV 4 GI 90] AV 2 GI 90 AV 2 GD 90 AV 3]

> AV 10

> FIN

Ahora definimos los que van a dibujar los puntos de salida de cada calle.

El LOGO puede leer caracteres de cualquier periférico de entrada.

EXPERIENCIA Y PRACTICAS EN LOGO

Con SALIDA situamos a la Tortuga en cada punto en donde tiene que dibujar las salidas y llamamos al procedimiento SAL:

```
? PARA SALIDA
> SL
> PONPOS [99 -28]
> SAL
> PONPOS [99 -8]
> SAL
> SL
> PONPOS [99 12]
> SAL
> SL
> PONPOS [99 32]
> SAL
> SL
> PONPOS [99 52]
> SAL
> FIN
```

SAL dibuja la salida:

```
? PARA SAL
> BL
> PONRUMBO -90
> REPITE 4 [BL AV 6 GD 90]
> PONRUMBO -45
> SL
> AV 2
> BL
> RELLENA
> PONRUMBO 90
> AV 4
> GI 90 AV 3
> GI 90 AV 2
> FIN
```

Una vez que tenemos todos los procedimientos definidos, los agrupamos en uno general:

```
? PARA PISCINA
> BP
> OT
> DIMEN
> CORCHERAS
> SALIDA
> FIN
```

Para ejecutarlo basta con introducir:

```
? PISCINA
```

4:

En primer lugar, definimos el procedimiento que va a dibujar la ficha del dominó, pero sin pintar en ella ningún punto:

```
? PARA FICHA
> SL
> PONPOS [-15 -20]
> BL
> REPITE 2 [AV 60 GD 90 AV 30 GD 90]
> AV 30 GD 90
> AV 30 RE 16
> GI 90 RE 1
> REPITE 4 [AV 2 GD 90]
> FIN
```

A continuación definimos otro procedimiento que va a dibujar simplemente un punto:

```
? PARA P
> BL
> REPITE 4 [AV 2 GD 90]
> FIN
```

Una vez que tenemos la ficha y el punto, definimos un procedimiento para cada uno de los números que aparecen en las fichas del dominó. Dependiendo del número que sea, dibujará los puntos correspondientes y en el lugar preciso.

En estos procedimientos vamos situando a la Tortuga en los sitios adecuados y realizando llamadas al procedimiento P para dibujar los puntos:

```
? PARA UNO
> SL
> PONX -1
> PONY -7 + :V
> P
> FIN
```

Al procedimiento que dibuja el dos lo llamamos DO, ya que algunas versiones de Logo no admite DOS como nombre de procedimiento, por ser una primitiva:

```
? PARA DO
> SL
> PONX -8
> PONY 1 + :V
> P
> SL
```

Por defecto, el periférico de donde LOGO recoge los caracteres es el teclado.

```
> PONX 6
> PONY -15 + :V
> P
> FIN
```

El tres se consigue dibujando el uno y el dos:

```
? PARA TRES
> UNO
> DO
> FIN
```

Para el cuatro podemos utilizar el dos y después dibujar los dos puntos que faltan:

```
PARA CUATRO
> DO
> SL
> PONX -8
> PONY -15 + :V
> P
> SL
> PONX 6
> PONY 1 + :V
> P
> FIN
```

El cinco lo conseguimos dibujando el cuatro y el uno:

```
? PARA CINCO
> CUATRO
> UNO
> FIN
```

Para el seis dibujamos primero el cuatro y a continuación los dos puntos que le faltan:

```
? PARA SEIS
> CUATRO
> SL
> PONX -8
> PONY -7 + :V
> P
> SL
> PONX 6
> PONY -7 + :V
> P
> FIN
```

La variable :V la utilizamos para dibujar, dependiendo de su valor, los números en la parte superior o en la inferior.

Una vez que tengamos todos los números, definimos un procedimiento para generar, en primer lugar, los números aleatorios. Estos

números son introducidos en las variables :A y :B.

La variable :A va a contener el número correspondiente al que hay que dibujar en la parte superior de la ficha y la variable :B el que se va a dibujar en la parte inferior. Una vez que los números son asignados a las variables, preguntamos, en primer lugar, por el contenido de la variable :A, y dependiendo del valor que tenga, realiza una llamada al procedimiento que dibuja el número correspondiente. Mientras que se están realizando las preguntas a la variable :A, la variable :V tiene asignado el valor 30. Este valor es el que hay que sumar a la coordenada Y en cada uno de los procedimientos que dibujan los números, para que éstos se dibujen en la parte superior de la ficha.

Una vez que se ha dibujado el número superior, se realiza el mismo proceso con la variable :B para dibujar el número inferior. En este caso a la variable :V se le asigna un 0.

```
? PARA CONTROL
> FICHA
> HAZ "A AZAR 7
> HAZ "B AZAR 7
> HAZ "V 30
> SI :A = 1 [ UNO ]
> SI :A = 2 [ DO ]
> SI :A = 3 [ TRES ]
> SI :A = 4 [ CUATRO ]
> SI :A = 5 [ CINCO ]
> SI :A = 6 [ SEIS ]
> HAZ "V 0
> SI :B = 1 [ UNO ]
> SI :B = 2 [ DO ]
> SI :B = 3 [ TRES ]
> SI :B = 4 [ CUATRO ]
> SI :B = 5 [ CINCO ]
> SI :B = 6 [ SEIS ]
> FIN
```

Ahora definimos el último procedimiento para que el proceso se realice 20 veces y así obtener en la pantalla el dibujo de 20 fichas del dominó:

```
? PARA VARIAS
> REPITE 10 [ BP CONTROL ESPERA
50 ]
> FIN
```

Si lo ejecutas, puedes obtener fichas como éstas:

Podemos saber si se ha pulsado o no una tecla.

EXPERIENCIA Y PRACTICAS EN LOGO

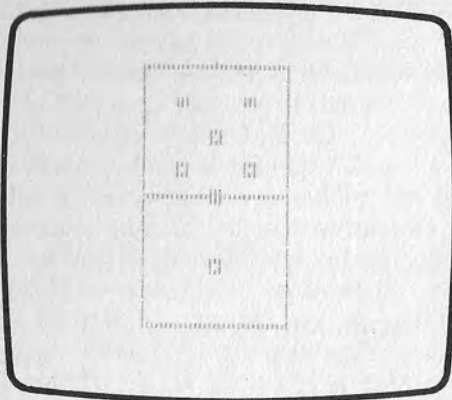


Fig. 2.

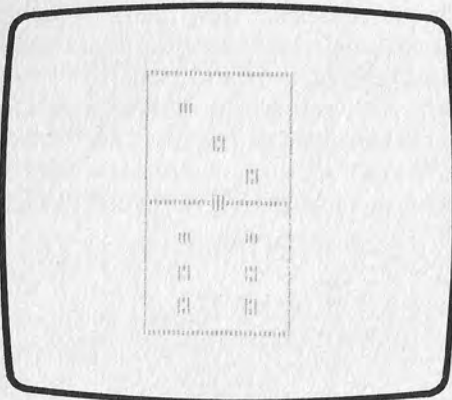


Fig. 3.

5:

```

— ? HAZ "A 1
  ? HAZ "B 0
  ? SI :A = 1 [ SI :B > 10 [ HAZ "A :B ] ]
CORRECTO

```

Si la condición (SI :A = 1) se cumple, se ejecutará la acción que hay dentro de los corchetes que a su vez es otra condición con una acción a realizar en el caso que se cumpla.

```

— ? ESCRIBIR LEECAR = :A
INCORRECTO

```

La variable A no tiene valor.

```

— ? PARA A
  > AV 10 GD 90
  > REPITE 3 [ AV 10 GD 120 ]
  > ALTO
  > FIN
CORRECTO

```

La orden ALTO puede aparecer en cualquier línea de un procedimiento y en solitario.

```

— ? HAZ "V AZAR 10
  ? SI :V > 2 ALTO [ RE :V ]
CORRECTO

```

En este caso la primitiva ALTO no es una acción del SI; por tanto, se cumpla o no la condición, la ejecución del procedimiento se detendrá.

```

— ? HAZ "A LEECAR
  ? ESPERA 25
  ? SI TECLA [ ESCRIBE :A + LEECAR ] [ ESCRIBE :A ]
INCORRECTO

```

LOGO devuelve el mensaje de error:

NO SE COMO DEBE HACER PARA TECLA.

Sería correcto:

```

? SI TECLA? [ ESCRIBE :A + LEECAR ] [ ESCRIBE :A ]
— ? HAZ "A 1
  ? SI TECLA? = :A [ ESCRIBE "CORRECTO ] [ ESCRIBE TECLA? ]
FALSO

```

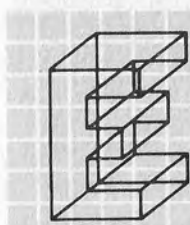
(FALSO resultado del SI.)

CORRECTO

La condición no se cumple, no hemos pulsado ninguna tecla antes de la comprobación; por tanto, se ejecutará la segunda acción.

- **La orden LEECAR interrumpe la ejecución del procedimiento en donde se encuentra.**
- **Con el LOGO no todo son dibujos.**
- **La primitiva del LOGO que lee un carácter no muestra nada en la pantalla.**

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



N el tomo anterior estuvimos viendo cómo tendríamos que hacer nuestros programas de juegos para que incluyesen disparos de balas. Vimos un programa para el SPECTRUM, y otro para el resto de los ordenadores. En

el primero de ellos un pistolero tenía que disparar a una serie de bloques que aparecían aleatoriamente en la pantalla. En el segundo el que disparaba era un tanque. Como se pudo apreciar, mientras la bala estaba moviéndose, ni el pistolero ni el tanque podían moverse por la pantalla. Ese es un error que no pueden tener nunca los programas que nosotros mismos realicemos. Es muy importante que mientras la bala se mueve nosotros también nos podamos mover por la pantalla.

Al final del tomo anterior se le dijo al

lector que probase y que intentase realizar un programa que permitiese el movimiento de la bala a la vez que el del pistolero o el del tanque. También se pedía que al chocar la bala con uno de los cuadrados éste explotase. Lógicamente, mientras cualquiera de los cuadrados está explotando, el hombre (o el tanque) tienen que seguir pudiéndose mover.

Los dos programas que vamos a ver a continuación cumplen perfectamente estos cometidos y algunos más. Lo que no hacen es que la bala rebote si no choca con ningún cuadrado durante su movimiento. Se ha preferido dejar esta modificación en manos del lector para que coja práctica y cree sus propias rutinas, así como su propio estilo de programación.

El primero de los programas es una modificación del LLANERO SOLITARIO, que ya hemos visto.

```
100 REM *****
110 REM * EL LLANERO SOLITARIO *
120 REM *****
130 REM
140 CLS
150 REM
160 REM *** DEFINICION DEL LLANERO SOLITARIO ***
170 REM
180 DATA 0,0,60,60,255,60,62,60
190 DATA 24,60,126,127,127,126,126
200 DATA 0,0,0,0,128,254,120,0
210 DATA 126,126,102,102,102,102,119
220 REM
230 REM *** LECTURA DE LAS DATAS ***
240 REM
250 FOR I=0 TO 31
260   READ N
270   POKE USR "A"+I,N
280 NEXT I
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
290 REM
300 REM *** CREACION DE LOS CARACTERES DE LA EXPLOSION ***
310 REM
320 FOR I=32 TO 63
330   LET N=INT(RND*255)+1
340   POKE USR "A"+I,N
350 NEXT I
360 REM
370 REM *** COLOCACION DE LOS CUADRADITOS ***
380 REM
390 DIM A(22,32)
400 REM
410 INK 0
420 BORDER 1
430 PAPER 6
440 CLS
450 REM
460 FOR I=1 TO 5
470   LET C=INT(RND*5+1)
480   LET XT=INT(RND*20)+1
490   LET YT=INT(RND*16)+15
500   PRINT INK C;AT XT,YT;CHR$(143)
510   LET A(XT,YT)=1
520 NEXT I
530 REM
540 REM *** INICIALIZACION DE VARIABLES ***
550 REM
560 LET N=0
570 LET X=11
580 LET Y=1
590 LET D=0
600 LET P=0
610 LET SW=0
620 REM
630 REM *** PROGRAMA PRINCIPAL ***
640 REM
650 PRINT AT X-1,Y;CHR$(144)
660 PRINT AT X,Y;CHR$(145);CHR$(146)
670 PRINT AT X+1,Y;CHR$(147)
680 LET A$=INKEY$
690 IF A$="" THEN LET A$="8":GOTO 730
700 IF CODE A$<>54 AND CODE A$<>55 AND CODE A$<>56 THEN GOTO 680
710 IF CODE A$<>56 THEN PRINT AT X-1,Y;" ";AT X,Y;" ";AT X+1,Y;" "
720 IF A$="8" AND SW=0 THEN LET XB=X:LET YB=Y+2:LET SW=1:GOTO 730
730 GOSUB VAL(A$)*100+100
740 GOSUB 1000
750 GOTO 650
800 REM
810 REM *** MOVIMIENTO HACIA ABAJO ***
820 REM
830 LET X=X+1
840 IF X>20 THEN LET X=20
850 RETURN
900 REM
910 REM *** MOVIMIENTO HACIA ARRIBA ***
920 REM
930 LET X=X-1
940 IF X<1 THEN LET X=1
950 RETURN
1000 REM
1010 REM *** DISPARO DEL LLANERO ***
1020 REM
1030 IF D=5 THEN GOTO 1200
1040 IF P<>0 THEN GOSUB 2000:RETURN
1050 IF SW=0 THEN RETURN
```

```

1060 PRINT AT XB,YB;" ."
1070 IF A(XB,YB+1)=1 THEN LET P=1:RETURN
1080 LET YB=YB+1
1090 IF YB>30 THEN PRINT AT XB,YX;" ":LET D=D+1:LET SW=0
1100 RETURN
1200 REM
1210 REM *** FIN DEL JUEGO ***
1220 REM
1230 PRINT AT 0,0;"EL JUEGO HA TERMINADO"
1240 PRINT "ADIOS ...."
1250 STOP
2000 REM
2010 REM *** EXPLOSION DEL CUADRADITO ***
2020 REM
2030 GOSUB 2000+100*P
2040 IF N=3 THEN GOTO 2500
2050 RETURN
2100 PRINT AT XB,YB+1;CHR$(148)
2110 LET P=2
2120 RETURN
2200 PRINT AT XB,YB+1;CHR$(149)
2210 LET P=3
2220 RETURN
2300 PRINT AT XB,YB+1;CHR$(150)
2310 LET P=4
2320 RETURN
2400 PRINT AT XB,YB+1;CHR$(151)
2410 LET P=1
2420 LET N=N+1
2430 RETURN
2500 PRINT AT XB,YB+1;" "
2510 LET P=0
2520 LET SW=0
2530 LET N=0
2540 LET A(XB,YB+1)=0
2550 RETURN

```

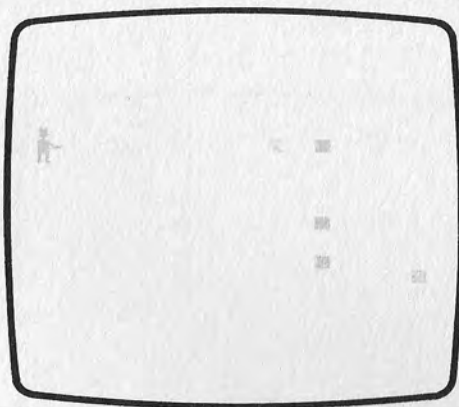


Fig. 1. Un momento después de haber chocado la bala con el cuadrado.

El segundo de los programas también es una versión mejorada del programa ATAQUE BELICO y que también vimos en el tomo anterior. Este programa está realizado bajo un IBM. Para que pueda ser utilizado en los demás ordenadores hay que realizar una serie de cambios.

```

100 REM *****
110 REM * ATAQUE BELICO *
120 REM *****
130 REM
140 CLS
150 RANDOMIZE TIMER
160 DIM A(24,40)
170 REM
180 REM *** SITUACION DE LOS BLANCOS A DISPARAR ***
190 REM
200 FOR I=1 TO 5
210 LET C=INT(RND*16)

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
220 IF C=2 THEN GOTO 210
230 LET XT=INT(RND*18)+2
240 LET YT=INT(RND*20)+20
250 COLOR C
260 LOCATE XT,YT
270 PRINT CHR$(177)
280 LET A(XT,YT)=1
290 NEXT I
300 REM
310 REM *** DISEÑO DEL TANQUE ***
320 REM
330 LET X=11
340 LET Y=2
350 LET D=0
360 COLOR 4
370 LOCATE X-2,Y+4
380 PRINT CHR$(218);CHR$(196)
390 LOCATE X-1,Y+4
400 PRINT CHR$(219);CHR$(219);STRING$(3,205)
410 LOCATE X,Y+1
420 PRINT STRING$(8,219)
430 LOCATE X+1,Y
440 PRINT STRING$(10,219)
450 REM
460 REM *** MOVIMIENTO Y DISPARO ***
470 REM
480 LET A$=INKEY$
490 IF A$="" THEN LET A$="Q":GOTO 600
500 IF A$<>"P" AND A$<>"L" AND A$<>"Q" THEN GOTO 480
510 IF A$="Q" AND SW=0 THEN LET XB=X-1:LET YB=Y+9:LET SW=1:GOTO 600
520 LOCATE X-2,Y+4
530 PRINT " "
540 LOCATE X-1,Y+4
550 PRINT SPC(5)
560 LOCATE X,Y+1
570 PRINT SPC(8)
580 LOCATE X+1,Y
590 PRINT SPC(10)
600 IF A$="Q" THEN GOSUB 660:GOTO 640
610 IF A$="L" THEN GOSUB 780
620 IF A$="P" THEN GOSUB 840
630 GOSUB 660
640 GOTO 370
650 REM
660 REM *** DISPARO DEL TANQUE ***
670 REM
680 IF D=5 THEN GOTO 900
690 IF P<>0 THEN GOSUB 960:RETURN
700 IF SW=0 THEN RETURN
710 LOCATE XB,YB
720 PRINT " ";CHR$(249)
730 IF A(XB,YB+1)=1 THEN LET P=1:RETURN
740 LET YB=YB+1
750 IF YB>38 THEN LOCATE XB,YB:PRINT " ":LET D=D+1:LET SW=0
760 RETURN
770 REM
780 REM *** MOVIMIENTO HACIA ABAJO ***
790 REM
800 LET X=X+1
810 IF X>20 THEN LET X=20
820 RETURN
830 REM
840 REM *** MOVIMIENTO HACIA ARRIBA ***
```

```

850 REM
860 LET X=X-1
870 IF X<3 THEN LET X=3
880 RETURN
890 REM
900 REM *** FIN DEL JUEGO ***
910 REM
920 LOCATE 1,1
930 PRINT "EL JUEGO HA TERMINADO."
940 PRINT "ADIOS ...."
950 END
960 REM
970 REM *** EXPLOSION DEL CUADRADITO ***
980 REM
990 ON P GOSUB 1020,1060,1100,1140
1000 IF N=3 THEN GOTO 1190
1010 RETURN
1020 LOCATE XB,YB+1
1030 PRINT CHR$(185)
1040 LET P=2
1050 RETURN
1060 LOCATE XB,YB+1
1070 PRINT CHR$(202)
1080 LET P=3
1090 RETURN
1100 LOCATE XB,YB+1
1110 PRINT CHR$(204)
1120 LET P=4
1130 RETURN
1140 LOCATE XB,YB+1
1150 PRINT CHR$(203)
1160 LET P=1
1170 LET N=N+1
1180 RETURN
1190 LOCATE XB,YB+1
1200 PRINT " "
1210 LET P=0
1220 LET SW=0
1230 LET N=0
1240 RETURN

```

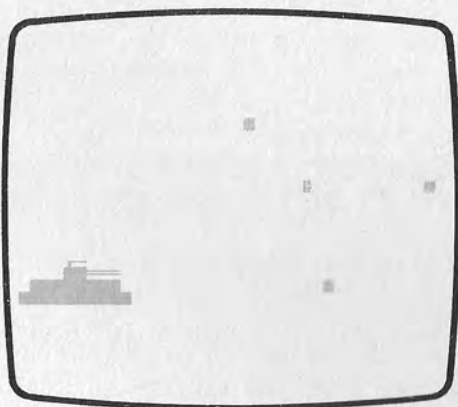


Fig. 2. Un cuadrado está explotando mientras el tanque se mueve hacia abajo.

COMMODORE

```

140 PRINT CHR$(147)
150 LET A=RND(-TI)
210 REM
230 LET XT=INT(RND(1)*18)+1
240 LET YT=INT(RND(1)*20)+20

```

```

250 REM
260 POKE 214,XT:POKE 211,YT
270 PRINT CHR$(166)
360 REM
370 POKE 214,X-2:POKE 211,Y+4
380 PRINT CHR$(176);CHR$(99)
390 POKE 214,X-1:POKE 211,Y+4
400 PRINT CHR$(166); CHR$(166); CHR$(168);
CHR$(168); CHR$(168)
410 POKE 214,X:POKE 211,Y+1
420 FOR Z=1 TO 8:PRINT CHR$(166);:NEXT Z
430 POKE 214,X+1:POKE 211,Y
440 FOR Z=1 TO 10:PRINT CHR$(166);:NEXT Z
480 GET A$
520 POKE 214,X-2:POKE 211,Y+4
540 POKE 214,X-1:POKE 211,Y+4
550 PRINT " " :REM 5 ESPACIOS
560 POKE 214,X:POKE 211,Y+1
570 PRINT " " :REM 8 ESPACIOS
580 POKE 214,X+1:POKE 211,Y
590 PRINT " " :REM 10 ESPACIOS
710 POKE 214,XB:POKE 211,YB
720 PRINT " "
750 IF YB>38 THEN POKE 214,XB:POKE 211,YB:PRINT
"LET D=D+1:LET SW=0
920 POKE 214,1:POKE 211,1

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1020 POKE 214,XB:POKE 211,YB+1
1030 PRINY CHR$(179)
1060 POKE 214,XB:POKE 211,YB+1
1070 PRINT CHR$(177)
1100 POKE 214,XB:POKE 211,YB+1
1110 PRINT CHR$(171)
1140 POKE 214,XB:POKE 211,YB+1
1150 PRINT CHR$(178)
1190 POKE 214,XB:POKE 211,YB+1
```

AMSTRAD

```
150 RANDOMIZE TIME
210 LET C=INT(RND(1)*7)+1
230 LET XT=INT(RND(1)*18)+2
240 LET YT=INT(RND(1)*20)+20
250 INK C
260 LOCATE YT,XT
270 PRINT CHR$(127)
360 INK 4
370 LOCATE Y+4,X-2
380 PRINT CHR$(150);CHR$(154)
390 LOCATE Y+4,X-1
400 PRINT CHR$(143);CHR$(143);STRING$(3,140)
410 LOCATE Y+1,X
420 PRINT STRING$(8,143)
430 LOCATE Y,X+1
440 PRINT STRING$(10,143)
520 LOCATE Y+4,X-2
540 LOCATE Y+4,X-1
560 LOCATE Y+1,X
580 LOCATE Y,X+1
710 LOCATE YB,XB
720 PRINT " ";CHR$(144)
750 IF YB>38 THEN LOCATE YB,XB:PRINT " ".LET
D=D+1:LET SW=0
1020 LOCATE YB+1,XB
1030 PRINT CHR$(158)
1060 LOCATE YB+1,XB
1070 PRINT CHR$(157)
1100 LOCATE YB+1,XB
1110 PRINT CHR$(155)
1140 LOCATE YB+1,XB
1150 PRINT CHR$(151)
1190 LOCATE YB+1,XB
```

MSX

```
150 LET A=RND(-TIME)
210 LET C=INT(RND(1)*7)+1
230 LET XT=INT(RND(1)*18)+2
240 LET YT=INT(RND(1)*20)+20
260 LOCATE YT,XT
270 PRINT CHR$(215)
370 LOCATE Y+4,X-2
380 PRINT CHR$(199);CHR$(195)
390 LOCATE Y+4,X-1
400 PRINT CHR$(219);CHR$(219);STRING$(3,195)
410 LOCATE Y+1,X
420 PRINT STRING$(8,219)
430 LOCATE Y,X+1
440 PRINT STRING$(10,219)
520 LOCATE Y+4,X-2
540 LOCATE Y+4,X-1
560 LOCATE Y+1,X
580 LOCATE Y,X+1
710 LOCATE YB,XB
720 PRINT " ";CHR$(188)
```

```
750 IF YB>38 THEN LOCATE YB,XB:PRINT " ".LET
D=D+1:LET SW=0
1020 LOCATE YB+1,XB
1030 PRINT CHR$(192)
1060 LOCATE YB+1,XB
1070 PRINT CHR$(198)
1100 LOCATE YB+1,XB
1110 PRINT CHR$(195)
1140 LOCATE YB+1,XB
1150 PRINT CHR$(201)
1160 LOCATE YB+1,XB
```

Vamos a ver ahora cómo están hechos estos dos programas. Como ambos son muy parecidos (lo único que varían son los números de línea y la forma de hacer las mismas cosas), sólo vamos a explicar el funcionamiento del primer programa. Si se entiende éste, se puede comprender perfectamente cómo funciona el segundo. La lógica, e incluso la forma de hacer las cosas, es igual.

Debido a que muchas de las partes son iguales a las de la versión anterior, sólo explicaremos aquéllas que son distintas. No hay que preocuparse porque los números de línea sean distintos en los programas. Da lo mismo en qué líneas se encuentren un grupo de instrucciones para que desarrollen una función.

De las partes que no vamos a comentar sólo diremos la función que realizan.

Líneas 180 a 210. En ellas se encuentra almacenado en líneas DATA la figura del LLA-NERO SOLITARIO.

Líneas 250 a 280. Leemos los números de las líneas DATAS y los introducimos en la zona de caracteres definibles.

Línea 320. Comenzamos un bucle dentro del cual vamos a definir las cuatro posiciones que configurarán la explosión de una cuadrado al ser alcanzado por una bala. La definición será aleatoria.

Línea 330. Obtenemos un número aleatorio entre 1 y 255 y lo almacenamos en la variable numérica N.

Línea 340. Almacenamos dicho número en la zona de memoria que guarda los caracteres definidos por el usuario. Los guardamos a continuación de los cuatro anteriores.

Línea 350. Aquí termina el bucle.

Línea 390. Definimos una matriz de 22 x 32 elementos. Esta nos servirá para saber cuándo la bala ha chocado con un cuadrado.

Línea 410 a 440. Ponemos los colores de la tinta, del papel y del borde. Borrarnos la pantalla.

Línea 460 a 520. Imprimimos de forma aleatoria por la pantalla cinco cuadraditos de colores también aleatorios. La única línea que

aparece como nueva en este trozo del programa es la 510. En ella almacenamos un uno en el elemento de la matriz que tiene el mismo número de fila y de columna que el cuadrado que acabamos de imprimir en la pantalla.

Línea 560. Ponemos la variable N a cero. Esta variable será la encargada de contar las veces que un cuadradito ha explotado antes de hacerlo desaparecer de la pantalla.

Líneas 570 y 580. Definimos la posición del pistolero en la pantalla.

Línea 590. Ponemos a cero la variable D. Esta nos indicará el número de balas que hemos disparado hasta el momento.

Línea 600. Ponemos a cero la variable B. Utilizaremos esta variable para saber qué momento de la explosión tenemos que imprimir la siguiente vez que tengamos que hacerlo.

Línea 610. Ponemos SW a cero. Esta variable nos indicará si hay una bala moviéndose por la pantalla.

Líneas 650 a 670. Imprimimos la figura del pistolero en la pantalla.

Línea 680. Leemos del teclado si el usuario ha pulsado alguna tecla.

Línea 690. Esta línea es muy importante. Si el usuario no ha pulsado ninguna tecla, entonces hacemos que A\$ sea igual al carácter '8'. Después nos vamos a la línea 730. Esta línea nos va a permitir que la bala se mueva, aunque nosotros no hayamos pulsado ninguna tecla. Más adelante esto quedará más claro.

Línea 720. Preguntamos la variable A\$ es igual al carácter '8' y si, a la vez, no hay ninguna bala moviéndose por la pantalla. A\$ será igual a '8' cuando el usuario pulse la tecla de disparo. Se comprueba que la variable SW es igual a cero para que no haya dos balas a la vez en la pantalla. En el caso de que la comprobación sea correcta, entonces almacenamos en XB la fila desde donde tiene que salir la bala, en YB la columna desde donde empieza el movimiento y ponemos SW a uno, puesto que vamos a empezar a mover una bala por la pantalla.

Línea 730. Nos vamos a la línea que nos interese.

Línea 740. Después de volver de la línea en la que nos encontrábamos hacemos un GOSUB 1000 por una razón muy importante. Si el usuario pulsó alguna tecla de movimiento, entonces no hemos movido la bala. Al mandar el control del programa a la línea 1000 vamos

a moverla una posición más hacia adelante antes de mirar si se está pulsando otra tecla.

Líneas 830 a 850. Se controla el movimiento hacia abajo del pistolero.

Líneas 930 a 950. Se controla el movimiento hacia arriba del pistolero.

Línea 1030. Si se han agotado todas las balas, nos vamos a la línea 1200 para imprimir un mensaje de despedida.

Línea 1040. Si hemos chocado con alguno de los cuadraditos ($P < > 0$), entonces nos vamos a la línea 2000 para gestionar la explosión. Cuando el ordenador vuelva de esta línea, retornamos al programa principal.

Línea 1050. Si no hay ninguna bala en camino, nos volvemos al programa principal.

Línea 1060. Imprimimos la bala en la siguiente posición de la pantalla.

Línea 1070. Y comprobamos que la siguiente posición no está ocupada por un cuadradito. En caso de que lo esté, ponemos la variable P a uno para que cuando el programa vuelva a pasar por esta rutina, no mueva la bala, sino que forme la explosión. También envolvemos el control al programa general.

Línea 1080. Sumamos uno a la posición de la bala para imprimirla en la siguiente vuelta.

Línea 1090. Si hemos llegado al final de la pantalla, entonces borramos la bala, sumamos uno al contador de balas gastadas y, como la bala ya ha desaparecido, ponemos a cero la variable SW. Esta será una de las líneas que tendrás que modificar sin quieres que la bala invierta su trayectoria al llegar al final de la pantalla sin chocar con ningún objeto.

Línea 1100. Nos volvemos al programa principal. Como se puede ver, sólo hemos movido la bala una posición hacia la derecha.

Líneas 1230 a 1250. Nos despedimos del jugador.

Línea 2030. Dependiendo del valor de la variable P, nos vamos a la línea 2100, 2200, 2300 ó 2400. Como tenemos cuatro posiciones distintas de la explosión, el valor de P irá variando desde uno (1) hasta cuatro (4). Con cada valor de la variable P imprimiremos uno de los momentos de la explosión.

Línea 2040. Si hemos repetido la serie de los cuatro caracteres para realizar la explosión durante tres veces, entonces nos vamos a la línea 2500 para borrar el cuadrado.

Línea 2050. Retornamos al programa general.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Líneas 2100 a 2120. Imprimimos la primera fase de la explosión, hacemos que P tenga valor dos (2) para que la siguiente vez se escriba la segunda fase de la explosión y retornamos al principio de la rutina.

Líneas 2200 a 2220. Imprimimos la segunda fase de la explosión, hacemos que P sea igual a tres (3) y retornamos a la rutina principal.

Líneas 2300 a 2320. Imprimimos la tercera fase de la explosión, cambiamos el valor de P a cuatro (4) y nos volvemos a la rutina principal.

Líneas 2400 a 2430. Imprimimos la cuarta y última fase de la explosión, volvemos a poner el valor de P a uno (1) para que se vuelva a repetir la secuencia, le sumamos uno al contador de explosiones (al final tenemos que realizar la explosión tres veces) y nos volvemos a la rutina principal.

Líneas 2500 a 2550. Llegamos a este grupo de líneas cuando hemos realizado la explosión durante tres veces. Borramos el cuadrado que ha explotado, ponemos la variable P a cero para indicar que no hay explosiones, ponemos SW a cero para indicar que no hay balas moviéndose por la pantalla, hacemos que el contador de explosiones sea igual a cero, borramos el cuadrado de la matriz donde almacenábamos las incidencias que ocurrían en la pantalla y nos volvemos al programa principal.

Con esto queda claro el funcionamiento del primer programa y, lógicamente, también del segundo.

Se vuelve a recomendar al lector que intente hacer y resolver sus propios programas. A los usuarios de ordenadores que no sean IBM se les recomienda que lean la explicación del funcionamiento del primer programa y apliquen lo que aprendan en entender el segundo.

TRUCOS Y RUTINAS BASICAS



A hemos dicho muchas veces que una de las partes más importantes de los programas es la presentación. Un programa puede ser aceptado o rechazado, dependiendo de lo agradable que sea para el usuario. Se puede decir que para que un programa sea bueno, es indispensable que su presentación sea buena. Pero no sólo vale presentar de una forma bonita los programas. Cada tipo de programa tiene que estar presentado de una manera. Así, pues, la forma que tendrá que tener un programa de gestión en la pantalla es muy distinta de la que tendría un juego.

En este tomo vamos a ver una serie de

rutinas que nos permitirán crear letras más grandes de las que tiene el ordenador. Estas letras serán ocho veces más grandes. Poder dar mensajes con letras más grandes de lo normal puede hacer que el usuario se dé cuenta de ciertos mensajes que, de otra manera, no vería. Este tipo de letras son buenas para aplicar en programas donde el usuario va a ser un niño. Con éstas se puede conseguir que el niño no tenga que hacer esfuerzos para leer la pantalla y que se interese por el programa que está utilizando.

Sin esperar más vamos a ver el primer programa. Con él podremos imprimir este tipo de letras con sólo mandarle una serie de parámetros.

```
6000 REM *****
6010 REM *
6020 REM * <<< IMPRESION DE CARACTERES GIGANTES >>> *
6030 REM *
6040 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA *
6050 REM * ----- *
6060 REM * *
6070 REM * A$ = MENSAJE A IMPRIMIR *
6080 REM * X = COLUMNA DONDE EMPEZARA EL MENSAJE *
6090 REM * Y = FILA DONDE SE COLOCARA EL MENSAJE *
6100 REM * *
6110 REM * VARIABLES USADAS POR LA RUTINA *
6120 REM * ----- *
6130 REM * *
6140 REM * N = POSICION DE LA DEFINICION DEL CARACTER *
6150 REM * A = VARIABLE QUE RECOGE EL DATO DE LAS LINEAS DATA *
6160 REM * I = VARIABLE INDICE DEL PRIMER BUCLE *
6170 REM * J = VARIABLE INDICE DEL SEGUNDO BUCLE *
6180 REM * *
6190 REM * LA RUTINA NO DEVUELVE NINGUN VALOR *
6200 REM * *
6210 REM *****
6220 REM
6230 FOR I=1 TO LEN(A$)
6240 RESTORE 8010
6250 LET N=ASC(MID$(A$,I,1))
6260 LET N=N-32
6270 LET N=N*8
```

TRUCOS Y RUTINAS BASICAS

```

6280   FOR J=1 TO N
6290     READ A
6300   NEXT J
6310   FOR J=1 TO 8
6320     READ A
6330     LOCATE Y+J-1,X
6340     IF A>=128 THEN LET A=A-128:PRINT CHR$(219);:GOTO 6360
6350     PRINT " ";
6360     IF A>=64 THEN LET A=A-64:PRINT CHR$(219);:GOTO 6380
6370     PRINT " ";
6380     IF A>=32 THEN LET A=A-32:PRINT CHR$(219);:GOTO 6400
6390     PRINT " ";
6400     IF A>=16 THEN LET A=A-16:PRINT CHR$(219);:GOTO 6420
6410     PRINT " ";
6420     IF A>=8 THEN LET A=A-8:PRINT CHR$(219);:GOTO 6440
6430     PRINT " ";
6440     IF A>=4 THEN LET A=A-4:PRINT CHR$(219);:GOTO 6460
6450     PRINT " ";
6460     IF A>=2 THEN LET A=A-2:PRINT CHR$(219);:GOTO 6480
6470     PRINT " ";
6480     IF A=1 THEN PRINT CHR$(219);:GOTO 6500
6490     PRINT " ";
6500     PRINT
6510   NEXT J
6520   LET X=X+8
6530   IF X>80 THEN LET X=1:LET Y=Y+8
6540 NEXT I
6550 RETURN
8000 REM
8010 REM *****
8020 REM *   LINEAS DE DATA CON LA *
8030 REM * DEFINICION DE TODOS LOS *
8040 REM * CARACTERES.           *
8050 REM *****
8060 REM
8070 DATA 0,0,0,0,0,0,0,0      :REM ESPACIO
8080 DATA 24,24,24,24,24,0,24,0 :REM !
8090 DATA 108,108,108,0,0,0,0,0 :REM "
8100 DATA 108,108,254,124,254,108,108,0 :REM #
8110 DATA 24,62,88,60,26,124,24,0 :REM $
8120 DATA 0,99,102,12,24,51,99,0 :REM %
8130 DATA 56,108,56,118,220,220,118,0 :REM &
8140 DATA 24,24,48,0,0,0,0,0 :REM '
8150 DATA 12,24,48,48,48,24,12,0 :REM (
8160 DATA 48,24,12,12,12,24,48,0 :REM )
8170 DATA 0,102,60,255,60,102,0,0 :REM *
8180 DATA 0,24,24,126,24,24,0,0 :REM +
8190 DATA 0,0,0,0,0,24,24,48 :REM ,
8200 DATA 0,0,0,126,0,0,0,0 :REM -
8210 DATA 0,0,0,0,0,24,24,0 :REM .
8220 DATA 6,12,24,48,96,192,128,0 :REM /
8230 DATA 124,198,206,214,230,198,124,0 :REM 0
8240 DATA 24,56,24,24,24,126,0 :REM 1
8250 DATA 60,102,6,60,96,102,126,0 :REM 2
8260 DATA 60,70,6,28,6,102,60,0 :REM 3
8270 DATA 24,56,88,152,254,24,60,0 :REM 4
8280 DATA 126,98,96,60,6,102,60,0 :REM 5
8290 DATA 60,102,96,124,102,102,60,0 :REM 6
8300 DATA 126,70,6,12,24,24,24,0 :REM 7
8310 DATA 60,102,102,60,102,102,60,0 :REM 8
8320 DATA 60,102,102,62,6,102,60,0 :REM 9
8330 DATA 0,0,24,24,0,24,24,0 :REM :
8340 DATA 0,0,24,24,0,24,24,48 :REM ;
8350 DATA 12,24,48,96,48,24,12,0 :REM <
8360 DATA 0,0,126,0,0,126,0,0 :REM =

```

8370 DATA	96, 48, 24, 12, 24, 48, 96, 0	: REM >
8380 DATA	60, 102, 6, 12, 24, 0, 24, 0	: REM ?
8390 DATA	124, 198, 222, 222, 222, 192, 124, 0	: REM @
8400 DATA	24, 60, 102, 102, 126, 102, 102, 0	: REM A
8410 DATA	252, 102, 102, 124, 102, 102, 252, 0	: REM B
8420 DATA	60, 102, 192, 192, 192, 102, 60, 0	: REM C
8430 DATA	248, 108, 102, 102, 102, 108, 248, 0	: REM D
8440 DATA	254, 98, 104, 120, 104, 98, 254, 0	: REM E
8450 DATA	254, 98, 104, 120, 104, 96, 240, 0	: REM F
8460 DATA	60, 102, 192, 192, 206, 198, 126, 0	: REM G
8470 DATA	102, 102, 102, 126, 102, 102, 102, 0	: REM H
8480 DATA	126, 24, 24, 24, 24, 24, 126, 0	: REM I
8490 DATA	30, 12, 12, 12, 204, 204, 120, 0	: REM J
8500 DATA	230, 102, 108, 120, 108, 102, 230, 0	: REM K
8510 DATA	240, 96, 96, 96, 98, 102, 254, 0	: REM L
8520 DATA	198, 238, 254, 214, 214, 198, 198, 0	: REM M
8530 DATA	198, 230, 246, 222, 206, 198, 198, 0	: REM N
8540 DATA	56, 108, 198, 198, 198, 108, 56, 0	: REM O
8550 DATA	252, 102, 102, 120, 96, 96, 240, 0	: REM P
8560 DATA	56, 108, 198, 198, 218, 204, 118, 0	: REM Q
8570 DATA	252, 102, 102, 124, 108, 102, 226, 0	: REM R
8580 DATA	60, 102, 96, 60, 6, 102, 60, 0	: REM S
8590 DATA	126, 90, 24, 24, 24, 24, 60, 0	: REM T
8600 DATA	102, 102, 102, 102, 102, 102, 60, 0	: REM U
8610 DATA	102, 102, 102, 102, 102, 60, 24, 0	: REM V
8620 DATA	198, 198, 198, 214, 254, 238, 198, 0	: REM W
8630 DATA	198, 108, 56, 56, 108, 198, 198, 0	: REM X
8640 DATA	102, 102, 102, 60, 24, 24, 60, 0	: REM Y
8650 DATA	254, 198, 140, 24, 50, 102, 254, 0	: REM Z
8660 DATA	60, 48, 48, 48, 48, 48, 60, 0	: REM [
8670 DATA	192, 96, 48, 24, 12, 6, 3, 1	: REM \
8680 DATA	60, 12, 12, 12, 12, 12, 60, 0	: REM]
8690 DATA	24, 60, 126, 24, 24, 24, 24, 0	: REM ^
8700 DATA	0, 0, 0, 0, 0, 0, 0, 255	: REM _
8710 DATA	0, 48, 24, 12, 0, 0, 0, 0	: REM '
8720 DATA	0, 0, 120, 12, 124, 204, 118, 0	: REM a
8730 DATA	224, 224, 124, 102, 102, 102, 188, 0	: REM b
8740 DATA	0, 0, 60, 102, 96, 102, 60, 0	: REM c
8750 DATA	28, 12, 124, 204, 204, 204, 118, 0	: REM d
8760 DATA	0, 0, 60, 102, 126, 96, 60, 0	: REM e
8770 DATA	28, 54, 48, 120, 48, 48, 120, 0	: REM f
8780 DATA	0, 0, 62, 102, 102, 62, 6, 124	: REM g
8790 DATA	224, 96, 108, 118, 102, 102, 230, 0	: REM h
8800 DATA	24, 0, 56, 24, 24, 24, 60, 0	: REM i
8810 DATA	2, 0, 14, 6, 6, 102, 102, 60	: REM j
8820 DATA	224, 96, 102, 108, 120, 108, 230, 0	: REM k
8830 DATA	56, 24, 24, 24, 24, 24, 60, 0	: REM l
8840 DATA	0, 0, 108, 254, 214, 214, 198, 0	: REM m
8850 DATA	0, 0, 216, 102, 102, 102, 102, 0	: REM n
8860 DATA	0, 0, 60, 102, 102, 102, 60, 0	: REM o
8870 DATA	0, 0, 220, 102, 102, 124, 96, 240	: REM p
8880 DATA	0, 0, 118, 204, 204, 124, 12, 30	: REM q
8890 DATA	0, 0, 216, 108, 96, 96, 240, 0	: REM r
8900 DATA	0, 0, 60, 96, 60, 6, 124, 0	: REM s
8910 DATA	48, 48, 124, 48, 48, 54, 28, 0	: REM t
8920 DATA	0, 0, 102, 102, 102, 102, 62, 0	: REM u
8930 DATA	0, 0, 102, 102, 102, 60, 24, 0	: REM v
8940 DATA	0, 0, 198, 214, 214, 254, 108, 0	: REM w
8950 DATA	0, 0, 198, 108, 56, 108, 198, 0	: REM x
8960 DATA	0, 0, 102, 102, 102, 62, 6, 124	: REM y
8970 DATA	0, 0, 126, 76, 24, 48, 126, 0	: REM z
8980 DATA	14, 24, 24, 112, 24, 24, 14, 0	: REM {
8990 DATA	24, 24, 24, 24, 24, 24, 24, 0	: REM
9000 DATA	112, 24, 24, 14, 24, 24, 112, 0	: REM }
9010 DATA	118, 216, 0, 0, 0, 0, 0, 0	: REM ~
9020 DATA	204, 51, 204, 51, 204, 51, 204, 51	: REM

TRUCOS Y RUTINAS BASICAS



Fig. 1. Resultado de ejecutar el programa 1.

Como siempre, para que el programa pueda funcionar en ordenadores distintos del IBM y compatibles, tendremos que hacer las siguientes modificaciones:

COMMODORE

```
6330 POKE 214,Y+J-1:POKE 211,X
6340 IF A>=128 THEN LET A=A-128:PRINT
CHR$(166)::GOTO 6360
6360 IF A>=64 THEN LET A=A-64:PRINT
CHR$(166)::GOTO 6380
6380 IF A>=32 THEN LET A=A-32:PRINT
CHR$(166)::GOTO 6400
6400 IF A>=16 THEN LET A=A-16:PRINT
CHR$(166)::GOTO 6420
6420 IF A>=8 THEN LET A=A-8:PRINT CHR$(166)::GO-
TO 6440
6440 IF A>=4 THEN LET A=A-4:PRINT CHR$(166)::GO-
TO 6460
6460 IF A>=2 THEN LET A=A-2:PRINT
CHR$(166)::GOTO 6480
6480 IF A=1 THEN PRINT CHR$(166)::GOTO 6500
```

SPECTRUM

```
6250 LET N=CODE(A$(I))
6330 PRINT AT Y+J-1,X;
6340 IF A>=128 THEN LET A=A-128:PRINT
CHR$(143)::GOTO 6360
6360 IF A>=64 THEN LET A=A-64:PRINT
CHR$(143)::GOTO 6380
6380 IF A>=32 THEN LET A=A-32:PRINT
CHR$(143)::GOTO 6400
6400 IF A>=16 THEN LET A=A-16:PRINT
CHR$(143)::GOTO 6420
6420 IF A>=8 THEN LET A=A-8:PRINT CHR$(143)::GO-
TO 6440
6440 IF A>=4 THEN LET A=A-4:PRINT CHR$(143)::GO-
TO 6460
6460 IF A>=2 THEN LET A=A-2:PRINT CHR$(143)::GO-
TO 6480
6480 IF A=1 THEN PRINT CHR$(143)::GOTO 6500
```

AMSTRAD

```
6330 LOCATE X,Y+J-1
6340 IF A>=128 THEN LET A=A-128:PRINT
CHR$(143)::GOTO 6360
```

```
6360 IF A>=64 THEN LET A=A-64:PRINT
CHR$(143)::GOTO 6380
6380 IF A>=32 THEN LET A=A-32:PRINT
CHR$(143)::GOTO 6400
6400 IF A>=16 THEN LET A=A-16:PRINT
CHR$(143)::GOTO 6420
6420 IF A>=8 THEN LET A=A-8:PRINT CHR$(143)::GO-
TO 6440
6440 IF A>=4 THEN LET A=A-4:PRINT CHR$(143)::GO-
TO 6460
6460 IF A>=2 THEN LET A=A-2:PRINT CHR$(143)::GO-
TO 6480
6480 IF A=1 THEN PRINT CHR$(143)::GOTO 6500
```

MSX

```
6330 LOCATE X,Y+J-1
```

Una vez que hemos metido el programa, como es muy fácil haberse equivocado al introducir las líneas DATA, tenemos que comprobar que todos los números son correctos. Para ello os proponemos introducir el programa 2. Con él podremos ver uno a uno todos los caracteres que hemos definido y apreciar si alguno de ellos no está definido correctamente.

```
100 REM *****
110 REM * PROGRAMA DEMO PARA VER QUE *
120 REM * LA TABLA DE CARACTERES HA *
130 REM * SIDO INTRODUCIDA CORRECTA- *
140 REM * MENTE. *
150 REM *****
160 REM
170 FOR C=32 TO 127
180 CLS
190 LET A$=CHR$(C)
200 PRINT "CARACTER No ";C
210 PRINT "CON FORMA = ";A$;" "
220 LET X=1
230 LET Y=10
240 GOSUB 360
250 PRINT
260 PRINT "PULSA 'P' PARA PARAR"
270 PRINT "OTRA PARA CONTINUAR"
280 LET A$=INKEY$
290 IF A$="" THEN GOTO 280
300 IF A$="P" OR A$="p" THEN STOP
310 NEXT C
320 CLS
330 PRINT "YA HEMOS VISTO TODOS"
340 PRINT "LOS CARACTERES."
350 END
```

Para que este programa funcione en el COMMODORE hay que realizar las siguientes modificaciones:

```
180 PRINT CHR$(147)
280 GET A$
320 PRINT CHR$(147)
```

En el SPECTRUM hay que cambiar la línea 350 por:

```
350 STOP
```

Si ya hemos comprobado que hemos introducido el programa sin errores o hemos corregido todos los errores que había, podemos ejecutar el programa 3 para experimentar un poco y ver cómo utilizar esta rutina.

```
100 REM *****
110 REM * PROGRAMA DEMO PARA EJECUTAR *
120 REM * LA Rutina DE LETRAS GRANDES *
130 REM *****
140 REM
150 CLS
160 INPUT "MENSAJE = ";A$
170 INPUT "COLUMNA = ";X
180 INPUT "FILAS = ";Y
190 PRINT
200 PRINT "PULSA UNA TECLA PARA EMPEZAR"
210 LET B$=INKEY$
220 IF B$="" THEN GOTO 210
230 CLS
240 GOSUB 6000
```

```
6310 FOR J=1 TO 8
6320 READ A
6330 LOCATE Y+J-1,X
6340 IF A>=128 THEN LET A=A-128:PRINT " ";:GOTO 6360
6350 PRINT CHR$(219);
6360 IF A>=64 THEN LET A=A-64:PRINT " ";:GOTO 6380
6370 PRINT CHR$(219);
6380 IF A>=32 THEN LET A=A-32:PRINT " ";:GOTO 6400
6390 PRINT CHR$(219);
6400 IF A>=16 THEN LET A=A-16:PRINT " ";:GOTO 6420
6410 PRINT CHR$(219);
6420 IF A>=8 THEN LET A=A-8:PRINT " ";:GOTO 6440
6430 PRINT CHR$(219);
6440 IF A>=4 THEN LET A=A-4:PRINT " ";:GOTO 6460
6450 PRINT CHR$(219);
6460 IF A>=2 THEN LET A=A-2:PRINT " ";:GOTO 6480
6470 PRINT CHR$(219);
6480 IF A=1 THEN PRINT " ";:GOTO 6500
6490 PRINT CHR$(219);
6500 PRINT
6510 NEXT J
```



Fig. 2. Podemos hacer que las letras aparezcan en inverso.

```
250 PRINT
260 PRINT "PULSA UNA TECLA PARA"
270 PRINT "INTRODUCIR OTRO MENSAJE"
280 LET B$=INKEY$
290 IF B$="" THEN GOTO 280
300 RUN
```

Para que este tercer programa se ejecute en el COMMODORE sin problemas hay que variar las siguientes líneas:

```
150 PRINT CHR$(147)
210 GET B$
230 PRINT CHR$(147)
280 GET B$
```

El funcionamiento del programa no lo veremos hasta el próximo tomo, debido a su extensión, pero vamos a ver algunas de las modificaciones que se le pueden hacer a este programa para que imprima las letras con distintos formatos y de distintas formas.

El programa 4, que aparece a continuación, nos permitirá imprimir las letras en inverso. Esto es, en vez de dibujar la letra, dibujaremos sus contornos.

Las modificaciones que hay que hacerle a este programa para que funcione en todos los ordenadores son:

COMMODORE

```
6330 POKE 214,Y+J-1:POKE 211,X
6350 PRINT CHR$(166);
6370 PRINT CHR$(166);
6390 PRINT CHR$(166);
6410 PRINT CHR$(166);
6430 PRINT CHR$(166);
```

TRUCOS Y RUTINAS BASICAS

```
6450 PRINT CHR$(166);
6470 PRINT CHR$(166);
6490 PRINT CHR$(166);
```

SPECTRUM

```
6330 PRINT AT Y+J-1,X;
6350 PRINT CHR$(143);
6370 PRINT CHR$(143);
6390 PRINT CHR$(143);
6410 PRINT CHR$(143);
6430 PRINT CHR$(143);
6450 PRINT CHR$(143);
6470 PRINT CHR$(143);
6490 PRINT CHR$(143);
```

AMSTRAD

```
6330 LOCATE X,Y+J-1
6350 PRINT CHR$(143);
```

```
6370 PRINT CHR$(143);
6390 PRINT CHR$(143);
6410 PRINT CHR$(143);
6430 PRINT CHR$(143);
6450 PRINT CHR$(143);
6470 PRINT CHR$(143);
6490 PRINT CHR$(143);
```

MSX

```
6330 LOCATE X,Y+J-1
```

Si queremos que, en vez de dibujar las letras con caracteres totalmente rectangulares y rellenos de forma, éstas se dibujen utilizando otro carácter (por ejemplo, un asterisco), podemos variar el programa para que lo haga. El resultado (con asteriscos) lo podemos ver en el programa 5:

```
6310 FOR J=1 TO 8
6320 READ A
6330 LOCATE Y+J-1,X
6340 IF A>=128 THEN LET A=A-128:PRINT "*";:GOTO 6360
6350 PRINT " ";
6360 IF A>=64 THEN LET A=A-64:PRINT "*";:GOTO 6380
6370 PRINT " ";
6380 IF A>=32 THEN LET A=A-32:PRINT "*";:GOTO 6400
6390 PRINT " ";
6400 IF A>=16 THEN LET A=A-16:PRINT "*";:GOTO 6420
6410 PRINT " ";
6420 IF A>=8 THEN LET A=A-8:PRINT "*";:GOTO 6440
6430 PRINT " ";
6440 IF A>=4 THEN LET A=A-4:PRINT "*";:GOTO 6460
6450 PRINT " ";
6460 IF A>=2 THEN LET A=A-2:PRINT "*";:GOTO 6480
6470 PRINT " ";
6480 IF A=1 THEN PRINT "*";:GOTO 6500
6490 PRINT " ";
6500 PRINT
6510 NEXT J
```

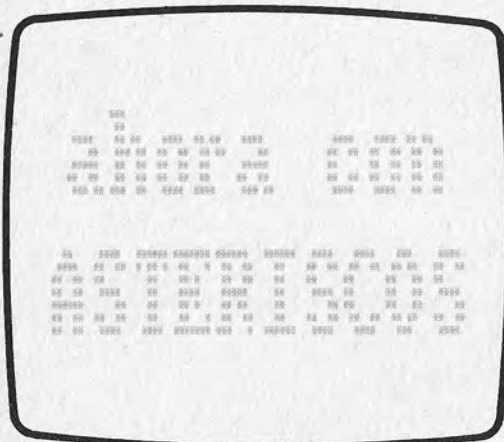


Fig. 3. Podemos cambiar el carácter que forma las letras.

Las únicas modificaciones que hay que hacer son las siguientes:

COMMODORE

```
6330 POKE 214,Y+J-1:POKE 211,X
```

SPECTRUM

```
6330 PRINT AT Y+J-1,X;
```

AMSTRAD Y MSX

```
6330 LOCATE X,Y+J-1
```

Por fin, si lo que queremos es dibujar las letras con el propio carácter que representan (ver figura 4), podemos variar el programa que lo haga de esta forma. El resultado lo vemos en el programa 6:

```

6230 FOR I=1 TO LEN(A$)
6240   RESTORE 9010
6250   LET N=ASC(MID$(A$,I,1))
6260   LET B$=MID$(A$,I,1)
6270   LET N=N-32
6280   LET N=N*8
6290   FOR J=1 TO N
6300     READ A
6310   NEXT J
6320   FOR J=1 TO 8
6330     READ A
6340     LOCATE Y+J-1,X
6350     IF A>=128 THEN LET A=A-128:PRINT B$;:GOTO 6370
6360     PRINT " ";
6370     IF A>=64 THEN LET A=A-64:PRINT B$;:GOTO 6390
6380     PRINT " ";
6390     IF A>=32 THEN LET A=A-32:PRINT B$;:GOTO 6410
6400     PRINT " ";
6410     IF A>=16 THEN LET A=A-16:PRINT B$;:GOTO 6430
6420     PRINT " ";
6430     IF A>=8 THEN LET A=A-8:PRINT B$;:GOTO 6450
6440     PRINT " ";
6450     IF A>=4 THEN LET A=A-4:PRINT B$;:GOTO 6470
6460     PRINT " ";
6470     IF A>=2 THEN LET A=A-2:PRINT B$;:GOTO 6490
6480     PRINT " ";
6490     IF A=1 THEN PRINT B$;:GOTO 6510
6500     PRINT " ";
6510     PRINT
6520   NEXT J
6530   LET X=X+8
6540   IF X>80 THEN LET X=1:LET Y=Y+8
6550 NEXT I
6560 RETURN

```



Fig. 4. Cada letra está formada por la misma letra que representa.

Las modificaciones que hay que realizar son las siguientes:

COMMODORE

6340 POKE 214,Y+J-1:POKE 211,X

SPECTRUM

6250 LET N=CODE(MID\$(A\$(I))

6260 LET B\$=A\$(I)

6340 PRINT AT Y+J-1,X

AMSTRAD Y MSX

6340 LOCATE X,Y+J-1

Como se puede apreciar, al ejecutar el programa en ciertas ocasiones resulta demasiado lento. Por ello, hemos preparado otro que es más rápido pero que necesita más memoria para poder funcionar:

```

6000 REM *****
6010 REM *
6020 REM * <<< IMPRESION DE CARACTERES GIGANTES >>>
6030 REM *
6040 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA
6050 REM * -----
6060 REM * |
6070 REM * A$ = MENSAJE A IMPRIMIR

```

TRUCOS Y RUTINAS BASICAS

```

6080 REM * X = COLUMNA DONDE EMPEZARA EL MENSAJE *
6090 REM * Y = FILA DONDE SE COLOCARA EL MENSAJE *
6100 REM * *
6110 REM * VARIABLES USADAS POR LA RUTINA *
6120 REM * ----- *
6130 REM * *
6140 REM * N = POSICION DE LA DEFINICION DEL CARACTER *
6150 REM * A = VARIABLE QUE RECOGE EL DATO DE LA MATRIZ *
6160 REM * I = VARIABLE INDICE DEL PRIMER BUCLE *
6170 REM * J = VARIABLE INDICE DEL SEGUNDO BUCLE *
6180 REM * N()= MATRIZ QUE ALMACENA LAS DEFINICIONES *
6190 REM * *
6200 REM * LA RUTINA NO DEVUELVE NINGUN VALOR *
6210 REM * *
6220 REM *****
6230 REM
6240 REM *** LECTURA DE LAS DATAS ***
6250 REM
6260 DIM D(96,8)
6270 FOR I=1 TO 96
6280     FOR J=1 TO 8
6290         READ D(I,J)
6300     NEXT J
6310 NEXT I
6320 RETURN
7000 REM
7010 REM *** RUTINA PRINCIPAL ***
7020 REM
7030 FOR I=1 TO LEN(A$)
7040     LET N=ASC(MID$(A$,I,1))
7050     LET N=N-31
7060     FOR J=1 TO 8
7070         LET A=D(N,J)
7080         LOCATE Y+J-1,X
7090         IF A>=128 THEN LET A=A-128:PRINT CHR$(219);:GOTO 7110
7100         PRINT " ";
7110         IF A>=64 THEN LET A=A-64:PRINT CHR$(219);:GOTO 7130
7120         PRINT " ";
7130         IF A>=32 THEN LET A=A-32:PRINT CHR$(219);:GOTO 7150
7140         PRINT " ";
7150         IF A>=16 THEN LET A=A-16:PRINT CHR$(219);:GOTO 7170
7160         PRINT " ";
7170         IF A>=8 THEN LET A=A-8:PRINT CHR$(219);:GOTO 7190
7180         PRINT " ";
7190         IF A>=4 THEN LET A=A-4:PRINT CHR$(219);:GOTO 7210
7200         PRINT " ";
7210         IF A>=2 THEN LET A=A-2:PRINT CHR$(219);:GOTO 7230
7220         PRINT " ";
7230         IF A=1 THEN PRINT CHR$(219);:GOTO 7250
7240         PRINT " ";
7250         PRINT
7260     NEXT J
7270     LET X=X+8
7280     IF X>80 THEN LET X=1:LET Y=Y+8
7290 NEXT I
7300 RETURN
8000 REM
8010 REM *****
8020 REM * LINEAS DE DATA CON LA *
8030 REM * DEFINICION DE TODOS LOS *
8040 REM * CARACTERES. *
8050 REM *****
8060 REM
8070 DATA 0,0,0,0,0,0,0,0 :REM ESPACIO
8080 DATA 24,24,24,24,24,0,24,0 :REM !
8090 DATA 108,108,108,0,0,0,0,0 :REM "

```

8100 DATA	108, 108, 254, 124, 254, 108, 108, 0	:REM #
8110 DATA	24, 62, 88, 60, 26, 124, 24, 0	:REM \$
8120 DATA	0, 99, 102, 12, 24, 51, 99, 0	:REM %
8130 DATA	56, 108, 56, 118, 220, 220, 118, 0	:REM &
8140 DATA	24, 24, 48, 0, 0, 0, 0, 0	:REM '
8150 DATA	12, 24, 48, 48, 48, 24, 12, 0	:REM (
8160 DATA	48, 24, 12, 12, 12, 24, 48, 0	:REM)
8170 DATA	0, 102, 60, 255, 60, 102, 0, 0	:REM *
8180 DATA	0, 24, 24, 126, 24, 24, 0, 0	:REM +
8190 DATA	0, 0, 0, 0, 0, 24, 24, 48	:REM ,
8200 DATA	0, 0, 0, 126, 0, 0, 0, 0	:REM -
8210 DATA	0, 0, 0, 0, 0, 24, 24, 0	:REM .
8220 DATA	6, 12, 24, 48, 96, 192, 128, 0	:REM /
8230 DATA	124, 198, 206, 214, 230, 198, 124, 0	:REM 0
8240 DATA	24, 56, 24, 24, 24, 24, 126, 0	:REM 1
8250 DATA	60, 102, 6, 60, 96, 102, 126, 0	:REM 2
8260 DATA	60, 70, 6, 28, 6, 102, 60, 0	:REM 3
8270 DATA	24, 56, 88, 152, 254, 24, 60, 0	:REM 4
8280 DATA	126, 98, 96, 60, 6, 102, 60, 0	:REM 5
8290 DATA	60, 102, 96, 124, 102, 102, 60, 0	:REM 6
8300 DATA	126, 70, 6, 12, 24, 24, 24, 0	:REM 7
8310 DATA	60, 102, 102, 60, 102, 102, 60, 0	:REM 8
8320 DATA	60, 102, 102, 62, 6, 102, 60, 0	:REM 9
8330 DATA	0, 0, 24, 24, 0, 24, 24, 0	:REM :
8340 DATA	0, 0, 24, 24, 0, 24, 24, 48	:REM ;
8350 DATA	12, 24, 48, 96, 48, 24, 12, 0	:REM <
8360 DATA	0, 0, 126, 0, 0, 126, 0, 0	:REM =
8370 DATA	96, 48, 24, 12, 24, 48, 96, 0	:REM >
8380 DATA	60, 102, 6, 12, 24, 0, 24, 0	:REM ?
8390 DATA	124, 198, 222, 222, 222, 192, 124, 0	:REM @
8400 DATA	24, 60, 102, 102, 126, 102, 102, 0	:REM A
8410 DATA	252, 102, 102, 124, 102, 102, 252, 0	:REM B
8420 DATA	60, 102, 192, 192, 192, 102, 60, 0	:REM C
8430 DATA	248, 108, 102, 102, 102, 108, 248, 0	:REM D
8440 DATA	254, 98, 104, 120, 104, 98, 254, 0	:REM E
8450 DATA	254, 98, 104, 120, 104, 96, 240, 0	:REM F
8460 DATA	60, 102, 192, 192, 206, 198, 126, 0	:REM G
8470 DATA	102, 102, 102, 126, 102, 102, 102, 0	:REM H
8480 DATA	126, 24, 24, 24, 24, 24, 126, 0	:REM I
8490 DATA	30, 12, 12, 12, 204, 204, 120, 0	:REM J
8500 DATA	230, 102, 108, 120, 108, 102, 230, 0	:REM K
8510 DATA	240, 96, 96, 96, 98, 102, 254, 0	:REM L
8520 DATA	198, 238, 254, 214, 214, 198, 198, 0	:REM M
8530 DATA	198, 230, 246, 222, 206, 198, 198, 0	:REM N
8540 DATA	56, 108, 198, 198, 198, 108, 56, 0	:REM O
8550 DATA	252, 102, 102, 120, 96, 96, 240, 0	:REM P
8560 DATA	56, 108, 198, 198, 218, 204, 118, 0	:REM Q
8570 DATA	252, 102, 102, 124, 108, 102, 226, 0	:REM R
8580 DATA	60, 102, 96, 60, 6, 102, 60, 0	:REM S
8590 DATA	126, 90, 24, 24, 24, 24, 60, 0	:REM T
8600 DATA	102, 102, 102, 102, 102, 102, 60, 0	:REM U
8610 DATA	102, 102, 102, 102, 102, 60, 24, 0	:REM V
8620 DATA	198, 198, 198, 214, 254, 238, 198, 0	:REM W
8630 DATA	198, 108, 56, 56, 108, 198, 198, 0	:REM X
8640 DATA	102, 102, 102, 60, 24, 24, 60, 0	:REM Y
8650 DATA	254, 198, 140, 24, 50, 102, 254, 0	:REM Z
8660 DATA	60, 48, 48, 48, 48, 48, 60, 0	:REM [
8670 DATA	192, 96, 48, 24, 12, 6, 3, 1	:REM \
8680 DATA	60, 12, 12, 12, 12, 12, 60, 0	:REM]
8690 DATA	24, 60, 126, 24, 24, 24, 24, 0	:REM ^
8700 DATA	0, 0, 0, 0, 0, 0, 0, 255	:REM _
8710 DATA	0, 48, 24, 12, 0, 0, 0, 0	:REM '
8720 DATA	0, 0, 120, 12, 124, 204, 118, 0	:REM a
8730 DATA	224, 224, 124, 102, 102, 102, 188, 0	:REM b
8740 DATA	0, 0, 60, 102, 96, 102, 60, 0	:REM c

TRUCOS Y RUTINAS BASICAS

```

8750 DATA 28,12,124,204,204,204,118,0 :REM d
8760 DATA 0,0,60,102,126,96,60,0 :REM e
8770 DATA 28,54,48,120,48,48,120,0 :REM f
8780 DATA 0,0,62,102,102,62,6,124 :REM g
8790 DATA 224,96,108,118,102,102,230,0 :REM h
8800 DATA 24,0,56,24,24,24,60,0 :REM i
8810 DATA 2,0,14,6,6,102,102,60 :REM j
8820 DATA 224,96,102,108,120,108,230,0 :REM k
8830 DATA 56,24,24,24,24,24,60,0 :REM l
8840 DATA 0,0,108,254,214,214,198,0 :REM m
8850 DATA 0,0,216,102,102,102,102,0 :REM n
8860 DATA 0,0,60,102,102,102,60,0 :REM o
8870 DATA 0,0,220,102,102,124,96,240 :REM p
8880 DATA 0,0,118,204,204,124,12,30 :REM q
8890 DATA 0,0,216,108,96,96,240,0 :REM r
8900 DATA 0,0,60,96,60,6,124,0 :REM s
8910 DATA 48,48,124,48,48,54,28,0 :REM t
8920 DATA 0,0,102,102,102,102,62,0 :REM u
8930 DATA 0,0,102,102,102,60,24,0 :REM v
8940 DATA 0,0,198,214,214,254,108,0 :REM w
8950 DATA 0,0,198,108,56,108,198,0 :REM x
8960 DATA 0,0,102,102,102,62,6,124 :REM y
8970 DATA 0,0,126,76,24,48,126,0 :REM z
8980 DATA 14,24,24,112,24,24,14,0 :REM {
8990 DATA 24,24,24,24,24,24,24,0 :REM !
9000 DATA 112,24,24,14,24,24,112,0 :REM }
9010 DATA 118,216,0,0,0,0,0,0 :REM ~
9020 DATA 204,51,204,51,204,51,204,51 :REM

```

Las modificaciones son las mismas que hay en el programa 1, pero en líneas distintas. El lector no tiene por qué tener ninguna dificultad al realizarlas él solo.

Este último programa es más largo porque almacena la definición de los caracteres en una tabla. Por ello, antes de ejecutar la rutina para que imprima algún mensaje, hay que inicializar la rutina. Esto se consigue haciendo una llamada a la línea 6000. Esta llamada sólo hay que hacerla una vez y una vez hecha podemos llamar todas las veces que queramos a la rutina de impresión que se encuentra a partir de la línea 7000.

Si queremos utilizar el programa 3 en conjunción con esta última rutina, tendremos que cambiar las líneas 240 y 300 por:

```

240 GOSUB 7000
300 GOTO 150

```

e intercalar las líneas:

```

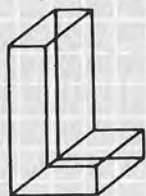
143 PRINT "ESPERE UN MOMENTO"
145 GOSUB 6000

```

En el tomo siguiente veremos el funcionamiento de estas rutinas y de algunas que nos permitirán hacer nuevos efectos con las letras.

EL TALLER DEL HARDWARE

■ Generador de señales de baja frecuencia



A generación de señales de prueba de equipos, empleando el circuito integrado LM555, se presentó en uno de los primeros tomos, como herramienta imprescindible del taller de hardware. Las señales genera-

das eran de forma rectangular, con la posibilidad de controlar la frecuencia, el ancho y la amplitud del pulso. Conviene disponer de un generador de señales más variadas, para analizar el comportamiento de los sistemas en condiciones más reales.

En el tomo anterior se ha descrito un sistema para poder analizar equipos de audio frecuencia de forma automática, utilizando el ordenador personal y algunos dispositivos adicionales para generar una señal y medir la respuesta del equipo ensayado. El componente más importante utilizado era un oscilador controlado por tensión en un solo circuito integrado. Este componente presenta un gran interés por las muchas aplicaciones en que puede utilizarse.

Vamos a describir otras aplicaciones del circuito ICL8038 como herramienta del taller de hardware y que podremos utilizar de forma autónoma o en conexión con el ordenador personal, para realizar ensayos en nuestros diseños, cuando las señales estén comprendidas en el rango de frecuencias que pueden generarse con dicho circuito.

Empezaremos con una descripción detallada del circuito, siguiendo a continuación con un esquema práctico de generador de señales de baja frecuencia que lo utiliza como

componente principal. Por último, se presentarán otros circuitos alternativos para la generación de señales senoidales, para poder calibrar las señales producidas por el equipo diseñado.

Circuito ICL8038

Es un circuito muy popular para la generación de señales de referencia, por la facilidad con que se pueden controlar los parámetros y por la posibilidad de disponer simultáneamente de salidas de tipo rectangular, triangular y senoidal.

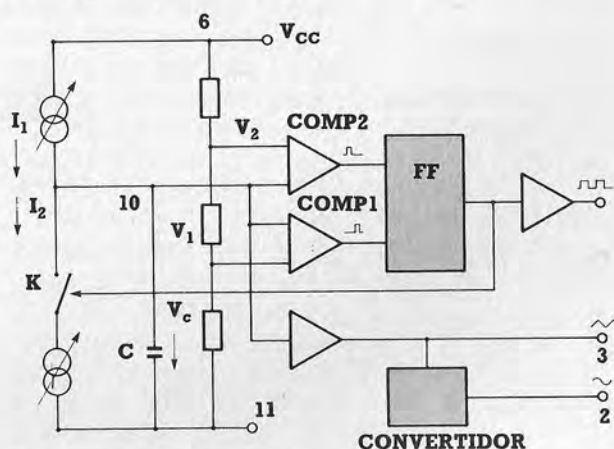


Fig. 1. Esquema de bloques del 8038.

El esquema recuerda al del circuito LM555, por la forma en que se obtiene la señal en rampa y la comparación con los niveles de referencia.

El funcionamiento básico es el siguiente:

Supongamos que se parte de una situación en la que el conmutador K se encuentra en la posición indicada en el esquema, es decir, abierto. La fuente de corriente de la parte

EL TALLER DEL HARDWARE

superior cargará el condensador C a intensidad constante I_1 . La tensión en el condensador crecerá de forma lineal, pues en un condensador la tensión es igual a la integral de la corriente, dividida por la capacidad. Es decir, la tensión es proporcional a la carga acumulada. Si en un momento dado se cierra el conmutador, las intensidades se modificarán según los valores de las fuentes de corriente. Con los valores indicados en el esquema, la nueva corriente resultante y que regulará la descarga del condensador será la diferencia entre las dos corrientes. Al ser también constante la tensión de descarga será también de forma lineal. Así podemos generar una señal triangular controlando los instantes de conmutación del conmutador K .

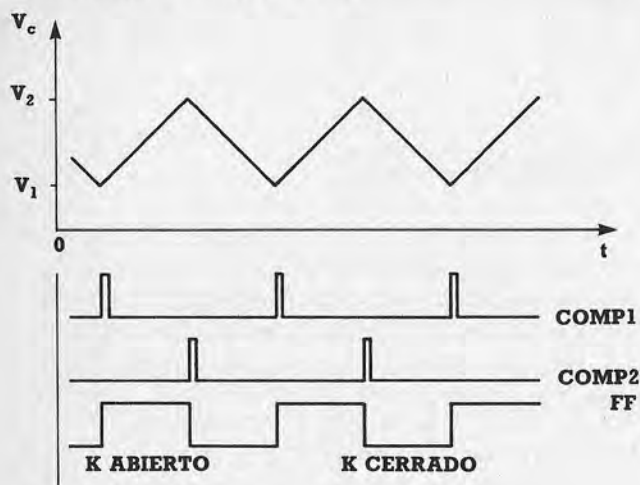


Fig. 2. Formas de onda en el circuito.

Para hacer el proceso operativo es necesario realizar la conmutación de forma automática, mediante las señales suministradas por los comparadores con las señales de referencia V_1 y V_2 . Estos comparadores COMP1, y COMP2 controlan la activación y desactivación del biestable FF, que es quien controla el conmutador K . La salida de COMP1 determinada por la comparación con V_1 , establece la apertura del conmutador K , mientras que la salida de COMP2 establece su cierre. El conmutador será realmente un transistor que pasa de estado de corte, cuando está abierto, a saturación, cuando está cerrado.

Por tanto, disponemos de los siguiente parámetros sobre los que influir para controlar la oscilación:

- Las corrientes de los generadores I_1 e I_2 .

- Los niveles de comparación V_1 y V_2 .
- El condensador C .

- La variación de estos parámetros determina la forma de empleo del circuito en cada caso, pero fácilmente se ve qué variaciones continuas de las corrientes y tensiones se corresponderán con variaciones correspondientes de los períodos de oscilación, mientras que variaciones por saltos discretos del condensador causarán saltos proporcionales a dicho salto.

Las señales directamente disponibles son de tipo triangular y rectangular, tomándose de los puntos indicados y amplificándose para su salida.

Generador de señal senoidal

La obtención de la señal senoidal es algo más complejo. Realmente no es una señal senoidal lo que se genera, sino una aproximación por tramos a partir de una señal triangular. Esto puede parecer sorprendente, pero es muy utilizado en los equipos electrónicos cuando se necesita una respuesta no lineal con relación a una magnitud: se linealiza por tramos. Cuanto más pequeño sea el tramo tanto mayor será la aproximación y, por tanto, la fidelidad de la señal generada con relación a la señal teórica. Veamos cómo puede realizarse:

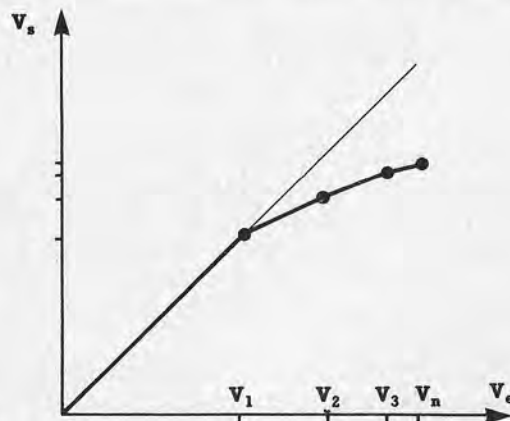


Fig. 3. Señal no lineal por tramos.

La señal de entrada ha de ser triangular para la generación de señales variables con el tiempo, pero puede ser de cualquier tipo en general para conversión con una escala no lineal. Las tensiones de las fuentes que se indican en el circuito las suponemos ideales, así como también los diodos, que presen-

tan un circuito abierto mientras la tensión aplicada a sus bornas sea negativa, y se comportan como un cortocircuito si la tensión aplicada es positiva.

Mientras la señal de entrada sea inferior a V_1 , todos los diodos estarán al corte, pues la tensión en sus bornas es negativa. Para tensiones de entrada por encima de V_1 , el diodo D_1 conducirá, con lo que la resistencia del circuito de salida será menor, reduciéndose la relación entre la tensión de entrada y la de salida. Para tensiones inferiores a V_1 la señal de salida será:

$$V_s = V_e \cdot R_B / (R_A + R_B)$$

Para tensiones superiores a V_1 y, por tanto, conduciendo D_1 la resistencia de salida será la equivalente a R_B en paralelo con R_1 , siendo consecuentemente menor la pendiente de la señal de salida.

Igualmente para los otros valores de las tensiones de ajuste se modificará la pendiente de la señal de salida. Lo mismo se puede hacer para tensiones negativas. El circuito se suele hacer simétrico, con los diodos puestos en conexión opuesta a los indicados en la figura y las tensiones de referencia serán negativas.

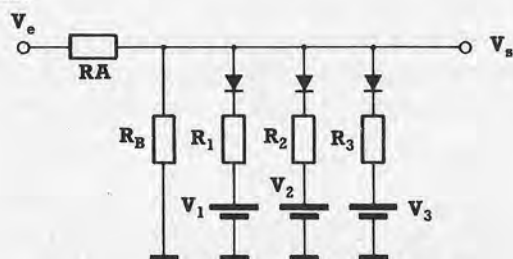


Fig. 4. Circuito de generación de señal no lineal.

Los diodos reales no se comportan como conmutadores perfectos, por lo que la transición de no conducción a conducción se efectúa de manera suave, y, en consecuencia, la salida será en realidad una curva con pequeñas transiciones en los puntos de cruce. Las tensiones de referencia pueden ser generadas mediante divisores resistivos, siempre que la corriente consumida sea muy baja. En el circuito real el esquema es algo más complicado, pues se utilizan transistores en seguidor de emisor, para presentar una gran impedancia de entrada al circuito de tensiones de referencia, generado por resistencias. En la figura se muestra el esquema real de la parte positiva del generador de senoide. La parte

negativa es la simétrica, utilizando los mismos valores de resistencias y los transistores de tipo complementario a los usados en la parte positiva, es decir, donde había un transistor de tipo PNP se coloca uno de tipo NPN, y al contrario. En paralelo con las resistencias extremas de la red resistiva de referencias puede colocarse una resistencia ajustable, con objeto de mejorar la simetría de la señal senoidal.

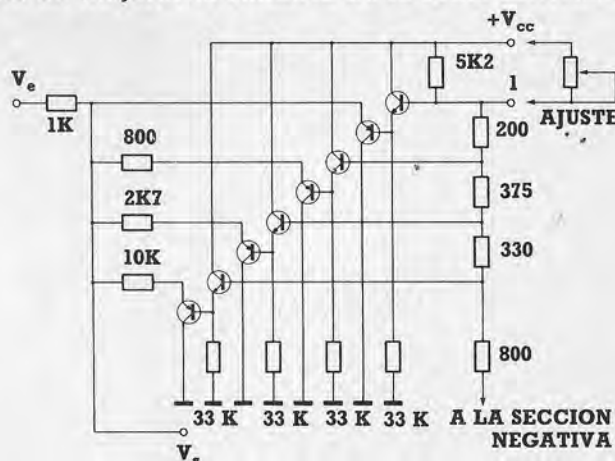


Fig. 5. Generación de senoide a partir de triangular.

Es interesante conocer por qué se genera la señal senoidal de esta manera tan complicada, existiendo otros medios para obtenerla que requieren menos componentes. La razón principal es que, por un lado, la generación de una señal de baja frecuencia senoidal requiere componentes voluminosos para una constante de tiempo elevada y, lo que es peor, requiere más de uno y de igual valor, dentro de un margen muy reducido de tolerancia. Por otro lado, la estabilidad de los circuitos montados depende de la sensibilidad a la temperatura principalmente y, por tanto, es difícil garantizarla. Utilizando un sistema digital, cuya estabilidad podría garantizarse mediante señales de referencia generadas con cristales de cuarzo, puede obtenerse la señal triangular y de ella deducirse la aproximación de señal senoidal, con la precisión deseada.

■ Fuentes de corriente

Las fuentes de corriente se realizan con transistores controlados por resistencias, que han de ser conectadas a las patillas del circuito, pero por ser accesible la entrada del generador, podremos igualmente emplear cual-

EL TALLER DEL HARDWARE

quier circuito que la genere y, por tanto, controlar mediante un equipo externo las características de la corriente de entrada. Esto permite controlar la frecuencia de oscilación con tensión, tal como hicimos en el sistema descrito en el tomo anterior.

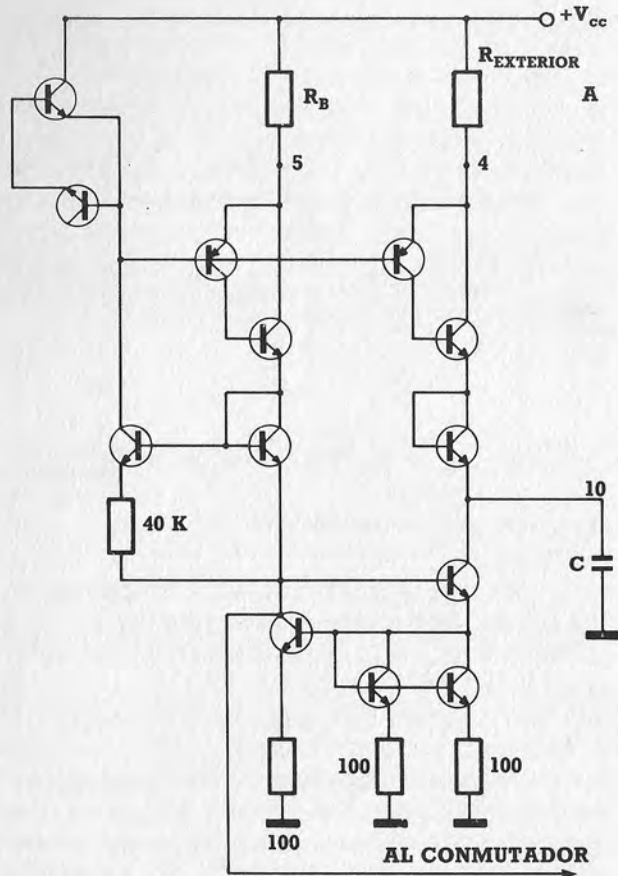


Fig. 6. Fuentes de corriente.

Comparadores

Los comparadores están realizados mediante amplificadores diferenciales con transistores. Los niveles de comparación son fijos, obtenidos mediante un divisor resistivo. Está realizado con tres resistencias iguales, que permiten obtener las dos tensiones de referencia de $1/3$ y $2/3$ de la tensión de alimentación. Esto permite una gran inmunidad a las variaciones de la tensión de alimentación, pues lo que realmente se utiliza es el valor relativo de los puntos de conmutación.

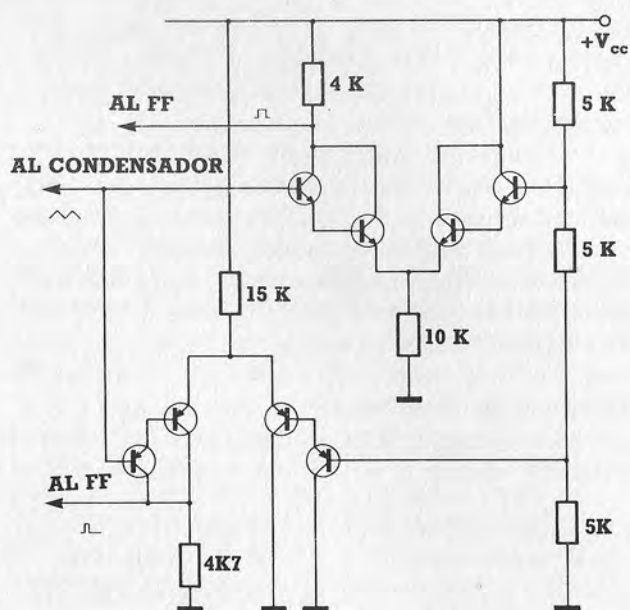


Fig. 7. Comparadores.

Biastable

El biastable de salida se realiza mediante transistores conectados con las bases cruzadas con los colectores opuestos.

Amplificadores de salida

Las señales de salida se obtienen con amplificadores con transistores complementarios para la señal triangular y en colector abierto para la salida rectangular, por lo que será necesario conectar una resistencia de carga conectada la tensión de alimentación.

Las señales de salida son proporcionales a la tensión de alimentación de acuerdo con los siguientes coeficientes:

- Señal rectangular = 0,90.
- Señal triangular = 0,33.
- Señal senoidal = 0,22.

Las amplitudes de salida son fijas, con una impedancia de salida entre 200 ohms y 2 Kohms. Debe utilizarse amplificador externo para garantizar la estabilidad.

Alimentación

La alimentación del circuito puede realizarse tanto de manera equilibrada, con tensiones positiva y negativa, como con tensión positiva y masa. En el primer caso, las señales de salida serán equilibradas respecto a la masa, mientras que con tensión positiva las señales triangular y senoidal resultan centradas respecto al valor medio de la tensión de alimentación.

El valor máximo de tensión de alimentación es de 30 voltios para tensión con referencia a masa y de ± 15 voltios para alimentación equilibrada. El valor mínimo de alimentación es del orden de 10 voltios o ± 5 para alimentación diferencial. Para nuestros equipos utilizaremos las tensiones de ± 12 que disponemos en el ordenador o en la fuente de alimentación de ampliación. Existen al menos cinco versiones del circuito integrado según el margen de temperaturas en que pueden trabajar y la estabilidad garantizada. Los de tipo comercial llevan como sufijo las letras CC, BC y AC y los de tipo militar BM y AM. El margen de temperatura de los primeros es de 0 a 70° C y el de los segundos de -55° a 125°.

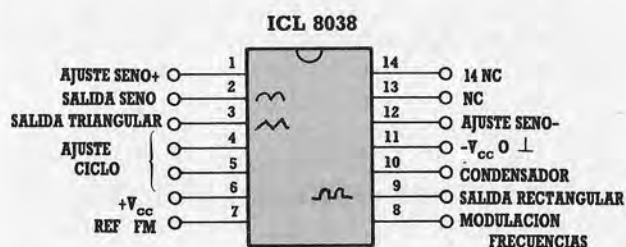


Fig. 8. Distribución de patillas.

Generador de señales práctico

Utilizando el circuito descrito y ya empleado en el tomo anterior, vamos a realizar un equipo que puede utilizarse de forma autónoma como instrumento del taller.

El generador de señales más simple requiere un número muy reducido de componentes, según se muestra en la figura.

La selección de la gama de frecuencias a generar se efectúa mediante un conmutador rotativo que conecta un condensador diferente para cada rango. La variación fina de fre-

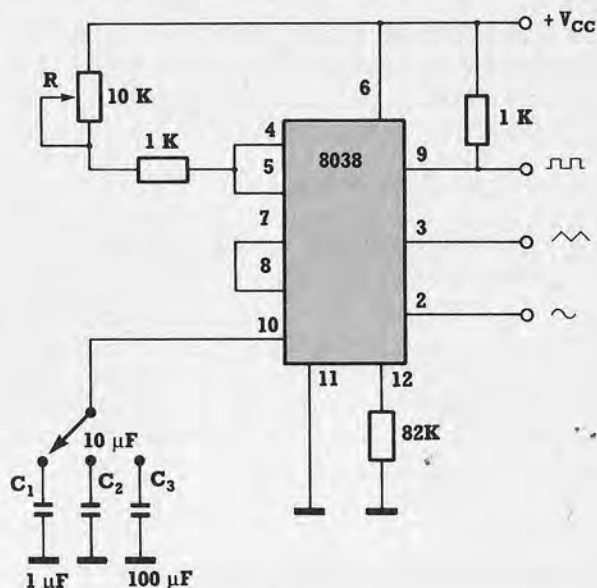


Fig. 9. Generador sencillo.

cuencia se efectúa mediante la resistencia variable R. El límite superior de frecuencia se fija por la resistencia fija colocada en serie con la resistencia variable.

En este esquema simple se han conectado las patillas 4 y 5 juntas, con lo que se pierde la posibilidad de ajuste de simetría de la señal.

El ajuste de distorsión, controlable a través de la patilla 12, se deja en posición fija, de acuerdo con el valor recomendado por el fabricante. En cualquier caso, el ajuste debería hacerse por comparación con una señal senoidal de referencia muy precisa, o mediante un distorsímetro, que es un aparato que resta una señal senoidal muy pura y de igual frecuencia que la de medida, y da como valor de la distorsión el valor eficaz de la diferencia. Nos conformaremos con el valor indicado.

La frecuencia de oscilación viene fijada por la fórmula:

$$F = 0,15/R \cdot C$$

La frecuencia resultará en Hz para los valores de resistencia en ohmios y de los condensadores en faradios. Los condensadores se elegirán en múltiplos de 10, para saltar una década en cada posición del conmutador rotativo. La resistencia variable ha de ser lo más lineal posible y de valor 10 veces el de la resistencia límite. Puede incluso tomarse un poco mayor para que el margen cubierto en cada escalón sea de 1 a 11, por ejemplo.

EL TALLER DEL HARDWARE

Puede ajustarse la simetría si se separa la conexión de las entradas de los generadores de corriente, de acuerdo con el esquema de la figura.

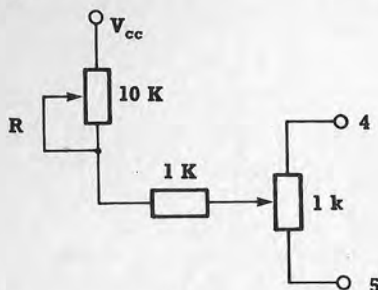


Fig. 10. Ajuste de simetría.

Generador de una sola banda

Es necesario a veces disponer de un generador que cubra un amplio margen de frecuencias de forma continua. Mediante el esquema de la figura puede verse cómo utilizando la entrada de tensión puede generarse toda la gama de frecuencias de audio.

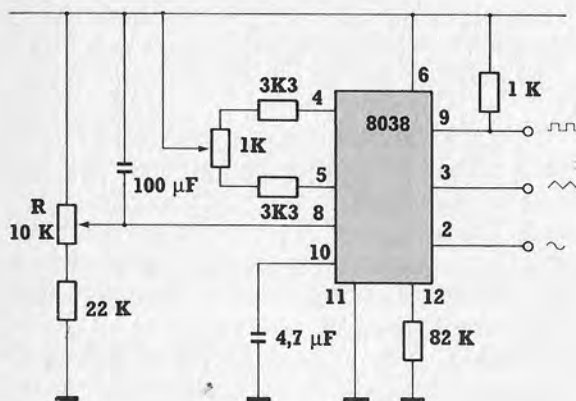


Fig. 11. Generador de audiofrecuencia de una sola banda.

Este circuito puede presentar cierta distorsión en la parte baja de frecuencias, pues no es posible el ajuste completo de la simetría de los generadores de corriente. Se incluye la resistencia de ajuste para reducirlo al mínimo.

Circuitos de salida

Para poder alimentar equipos a ensayar de tipo general es necesario amplificar las señales de salida. Para frecuencias bajas un am-

plificador basado en operacionales es suficiente para obtener el nivel necesario. Se le añade un potenciómetro para poder ajustar de forma continua el nivel de salida. A las frecuencias más elevadas puede aparecer el efecto de limitación en la rampa de subida del amplificador. Puede subsanarse el problema con un sencillo amplificador en seguidor de emisor, que presenta baja impedancia de salida.

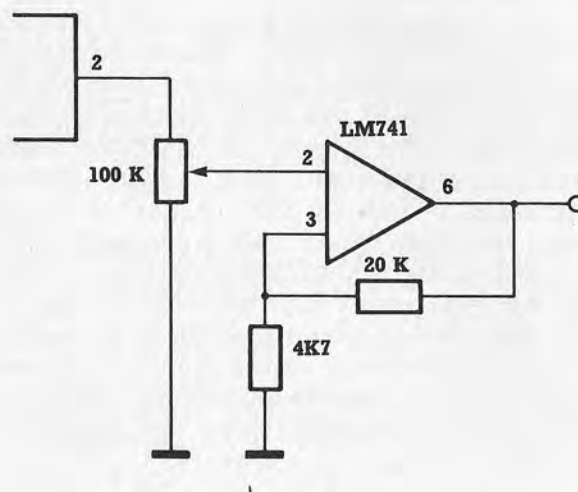


Fig. 12. Amplificador de salida.

Conexión al ordenador personal

Además del circuito mostrado en el tomo anterior, podemos utilizar el ordenador para controlar otras características del oscilador. Si se sustituyen los conmutadores por puertas analógicas o transistores MOS, podemos controlar con el puerto de salida de la tarjeta de ampliación de puertos el funcionamiento mediante programa. La variación de frecuencia mediante un conversor digital/analógico activando la entrada de la patilla 8 permite la generación de frecuencias variables con el código transmitido al conversor. También podemos generar trenes de señales de una frecuencia dada durante períodos programados. El esquema de la figura permite detener la señal de oscilación, cortocircuitando las bornas del condensador al hacerse conductor el transistor por efecto del pulso dado a su entrada.

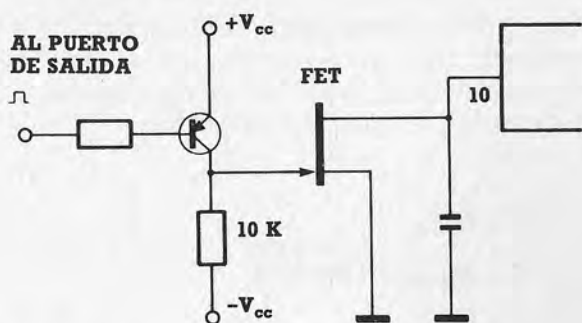


Fig. 13. Generación de ráfagas.

La generación de ráfagas puede hacerse simultáneamente a la variación de frecuencia, obteniéndose señales de prueba de equipos de audio-frecuencia más parecidas a las condiciones reales de trabajo.

Generador de señal senoidal de referencia

Con objeto de ajustar la simetría y conocer la posible influencia de los armónicos que indudablemente están presentes en el senoide generada mediante ajuste por tramos, podemos realizar un oscilador senoidal de frecuencia fija, dentro del margen de frecuencias de mayor interés en nuestras aplicaciones.

Un circuito típico de obtención de señal senoidal mediante amplificadores operacionales es el de puente de Wien. Consiste en establecer en el lazo de realimentación positiva un circuito que produzca las condiciones de

oscilación, que son el que la señal de realimentación se encuentre en fase con la de entrada al amplificador. Para que la oscilación sea estable, la ganancia del circuito ha de ser la unidad.

Al ser la atenuación en el lado de realimentación positiva de valor 3, es necesario que la ganancia sea también 3. Para que se mantenga estable es necesario utilizar un componente no lineal que atenúe la ganancia si la amplitud sobrepasa el valor deseado o que la aumente si es necesario mantener la oscilación. Los componentes no lineales utilizados generalmente son lámparas incandescentes miniatura, diodos zener o termistores. En cualquier caso el efecto observado es que la oscilación arranca, aumentando rápidamente la amplitud de la oscilación hasta que entra en funcionamiento el elemento no lineal y se produce el ajuste de ganancia.

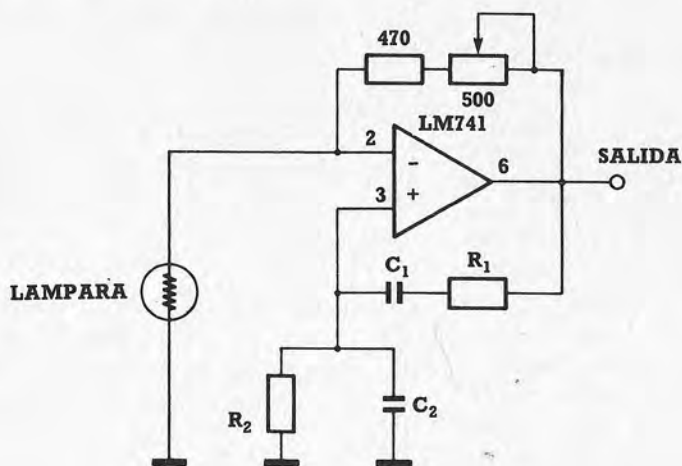
La frecuencia de oscilación viene fijada por:

$$f = 1 / 2\pi R C$$

Se suponen iguales las resistencias y condensadores: $R_1 = R_2 = R$, $C_1 = C_2 = C$.

El interés del circuito radica en que no es necesario utilizar bobinas, pues estos componentes son más difíciles de fabricar con la precisión y estabilidad adecuadas. La señal obtenida es de gran pureza y estabilidad, por lo que puede utilizarse como referencia para el otro tipo de oscilador.

Para hacer el oscilador de frecuencia variable hemos de montar dos resistencias variables sobre el mismo eje, para que se varíen a la vez.



$$F = \frac{1}{2\pi R_1 C_1}$$

$$R_1 = R_2$$

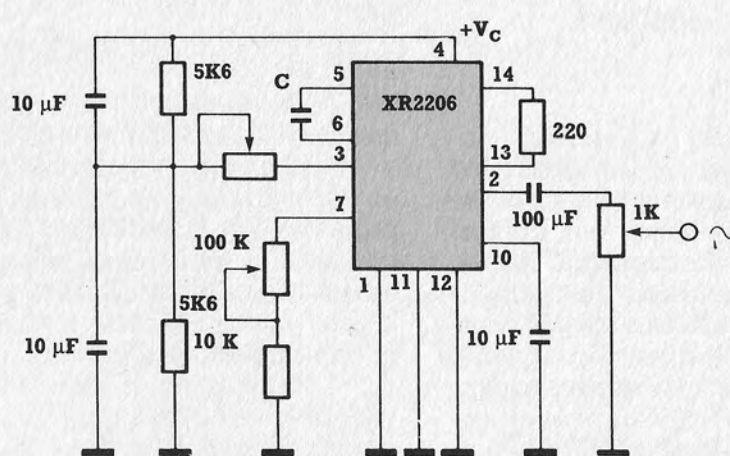
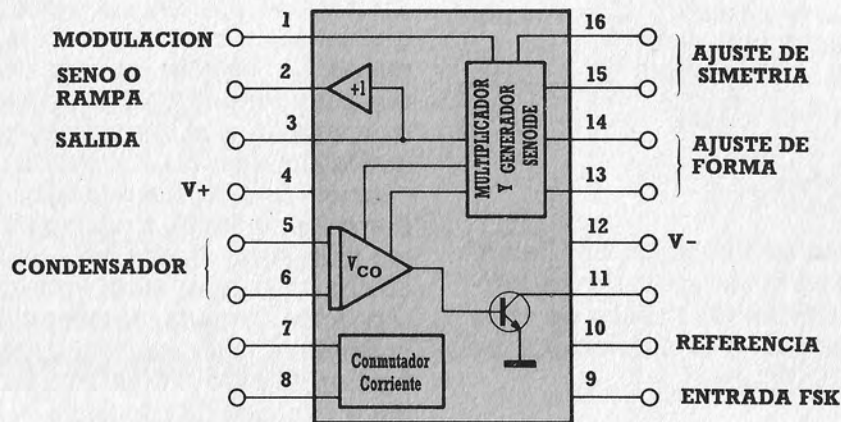
$$C_1 = C_2$$

Fig. 14. Oscilador senoidal en puente de Wien.

Otros circuitos osciladores

Existen otros circuitos similares al presentado para generar señales rectangulares,

triangulares y senoidales, también de fácil adquisición. Uno muy popular es el XR-2206, cuyo esquema se indica en la figura, junto con un circuito sencillo de aplicación.

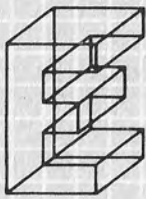


$$F = \frac{1}{2\pi RC}$$

Fig. 15. Circuito XR-2206 y oscilador senoidal.

NATURALEZA Y TECNOLOGIA

■ Péndulo simple



N general, todo el mundo tiene la idea de lo que es un péndulo como todo cuerpo que separado de su posición de equilibrio es susceptible de oscilar a un lado y a otro de dicha posición. Lo más cercano a nosotros que cumple estas condiciones es la lenteja que cuelga de un reloj de pared.

Sin embargo, en sentido más estricto, un péndulo es todo cuerpo pesado suspendido de un punto por encima de su centro de gravedad y que puede girar alrededor de un eje que pasa por dicho punto (Fig. 1).

Fig. 1a. Varilla vibrando. Fig. 1b. Paraguas.

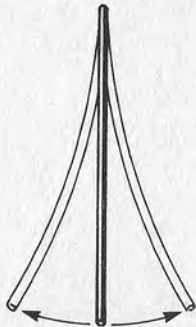


Fig. 1a. Varilla vibrando.

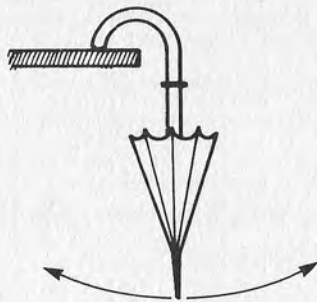


Fig. 1b. Paraguas.

Podemos encontrarnos con varios tipos de péndulo: *simple* y *compuesto*.

Vamos a centrarnos en el estudio del primero de ellos, el *péndulo simple*.

Péndulo simple

Se considera como péndulo simple a un punto material pesado suspendido en el vacío

por un hilo inextensible y sin peso, susceptible de oscilar sin rozamientos. Lo más aproximado a él a nivel de experimentación es una pequeña esfera de gran densidad, suspendida de un hilo fino y suficientemente largo (Fig. 2).

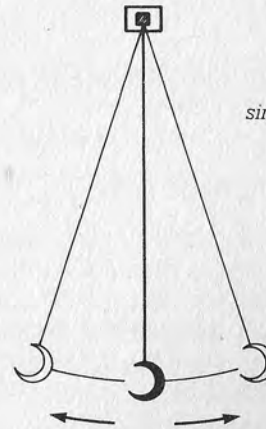


Fig. 2. Péndulo simple experimental.

Movimiento del péndulo simple. Ecuación del periodo T .

El movimiento del péndulo simple viene representado por la figura (Fig. 3):

Al apartar el peso, P , de su posición de equilibrio y llevarlo hasta el punto B , se ha realizado un trabajo que queda acumulado en el cuerpo como energía potencial.

Dejado libre, tenderá a bajar hacia la posición inicial, debido a la acción de su peso P , por la vertical (movimiento de caída libre), pero como está sujeto a un hilo inextensible, tiene que seguir el arco BA . Al peso se le puede descomponer en dos: una fuerza H y una fuerza N . La fuerza N (fuerza normal) queda anulada por la reacción del hilo y la fuerza H

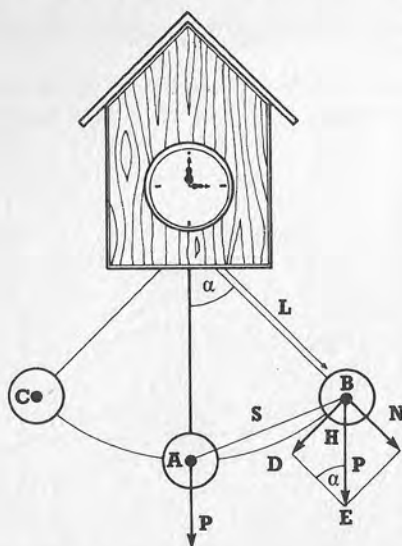


Fig. 3. Péndulo simple.

P = Peso.

H = Componente del peso que hace volver al cuerpo al punto «A» del equilibrio.

N = Componente del peso anulado por el hilo.

L = Longitud del brazo del péndulo

Triángulo OAB = Triángulo BDE

A = Punto de equilibrio.

es la que hace que el cuerpo se mueva hacia la posición A, cuyo valor es:

$$P = m \cdot g \quad /$$

$$H = P \cdot \sin \alpha / H = m \cdot g \cdot \sin \alpha \quad [1]$$

Al llegar al punto A ya no tiene energía potencial, pues ha dejado de recorrer una altura, pero por el teorema de la conservación de la energía (en un sistema aislado se mantiene constante la energía total del sistema, o sea, la suma de la energía potencial y la energía cinética se mantiene constante: E. cinética-1 + E. potencial-1 = E. cinética-2 + E. potencial-2), ha pasado todo a energía cinética, con lo que en el punto A ésta es máxima y ello hace que el cuerpo suba hasta el punto C (está a la misma altura que B). En este nuevo punto, al cumplirse también el teorema de la conservación de la energía, en el cuerpo toda la energía cinética se ha vuelto energía potencial. Y del punto C, por acción del peso, tenderá a bajar de nuevo al punto de equilibrio A. Este proceso de oscilamiento continuo seguirá indefinidamente si no fuera por el rozamiento.

Conocemos, por tanto, la fuerza H que el cuerpo oscile por [1]:

$$H = m \cdot g \cdot \sin \alpha \quad [1']$$

donde el valor del seno del ángulo, por trigonometría, sabemos que es:

$$\sin \alpha = \frac{s}{L} \quad [2]; H = m \cdot g \cdot \frac{s}{L} \quad [3]$$

Si la oscilación es muy pequeña, llega un momento en que el valor del $\sin \alpha$ y el valor de α se confunde, por lo que se puede poner:

$$H = m \cdot g \cdot \sin \alpha = m \cdot g \cdot \alpha = m \cdot g \cdot \frac{s}{L} \quad [3']$$

El movimiento del péndulo simple es un tipo de movimiento armónico, donde el valor de la fuerza toma el valor:

$$F = m \cdot w^2 \cdot s \quad [4']$$

Igualando ambas expresiones [3'] y [4], resulta:

$$mg \frac{s}{L} = mw^2 \cdot s \quad [4']$$

simplificando y despejando se obtiene:

$$g/L = w^2 \quad [5']$$

Por otra parte, en el movimiento armónico el valor de la velocidad angular es:

$$w = \frac{2\pi}{T} \quad [6]$$

de donde: $w^2 = \frac{(2\pi)^2}{T^2} \quad [6']$

Igualando [5] y [6']:

$$\frac{4\pi^2}{T^2} = \frac{g}{L} \quad [7]$$

simplificando y despejando:

$$T = 2\pi \cdot \sqrt{L/g} \quad [8]$$

que es el valor del período del péndulo simple, donde L corresponde al valor de la longitud del brazo del péndulo y g es la aceleración total del sistema y que según las condiciones será: 9,8 m/s² o bien g = 9,8 + a m/s² (a: aceleración del sistema si está en movimiento).

```

10 REM *****
15 REM *
20 REM *   EL PENDULO SIMPLE   *
25 REM *                       *
```

```

30 REM *****
40 CLS
50 LET PI=3.1416: LET G=9.8
55 REM ***** MENU PRINCIPAL *****
60 PRINT TAB(8); "PENDULO SIMPLE"
65 PRINT TAB(8); "-----"
70 PRINT:PRINT "PUEDE LLEGAR A DETERMINARSE EL"
72 PRINT "VALOR DEL PERIODO DE UN PENDULO"
76 PRINT "CONOCIENDO LA LONGITUD DEL BRAZO"
80 PRINT "Y LA ACELERACION DEL SISTEMA."
90 PRINT:PRINT
100 PRINT TAB(4); "ELIJE LAS CONDICIONES DEL"
105 PRINT "SISTEMA:"
110 PRINT : PRINT "1.- PENDULO EN CONDICIONES NOR-"
115 PRINT TAB(5); "MALES. (g = 9.8 m/s^2)"
120 PRINT:PRINT "2.- PENDULO EN UN OBJETO EN MO-"
125 PRINT TAB(5); "VIMIENTO. g.total= g+a."
126 PRINT TAB(4); "(a= aceleración del sistema)"
128 PRINT:PRINT: PRINT "(PULSA 1 / 2 ) ";
130 REM ***** ELECCION DE LA OPCION *****
140 INPUT U
150 ON U GOTO 300,500
160 CLS
170 REM ***** FIN DE PROGRAMA *****
190 PRINT:PRINT: PRINT "DESEAS CONOCER OTRO VALOR DEL"
195 PRINT "PERIODO <T> (S/N):";
198 INPUT A$
200 IF A$="S" OR A$="s" THEN CLS:GOTO 90
210 IF A$<>"N" AND A$<>"n" THEN GOTO 190
220 CLS: LOCATE 12,10:PRINT " !!! FIN DEL PROGRAMA !!! "
230 FOR I=1 TO 5000: NEXT I:CLS:END
300 REM *****
302 REM * INTRODUCCION DEL VALOR DEL BRAZO <L> DEL *
303 REM * PENDULO Y DETERMINACION DEL PERIODO <T> *
304 REM *****
306 CLS
308 PRINT:PRINT:PRINT
310 PRINT "LONGITUD DEL BRAZO DEL PENDULO"
315 PRINT "EN METROS: "
318 INPUT L
319 IF L=0 THEN CLS:PRINT "PERIODO DEL PENDULO <T>: T=0":FOR I=1 TO 3000:NEXT I:
GOTO 160
320 PRINT:PRINT:PRINT TAB(3); "EN ESTE CASO LA <g> EN LA FOR-"
325 PRINT "MULA DE <T> ES g = 9.8 m/s^2."
330 FOR I=1 TO 5000:NEXT I
400 REM *****
401 REM * CALCULO DEL PERIODO <T> DEL PENDULO *
402 REM * EN CONDICIONES NORMALES, G = 9.8 m/s^2. *
403 REM *****
405 CLS
410 LET T=2*PI*(L/G)^(1/2)
412 REM **** REDONDEO DEL VALOR DEL PERIODO A DOS DECIMALES ****
415 LET P=INT(T*100)/100
418 REM ***** PRESENTACION EN PANTALLA DEL VALOR DEL PERIODO *****
420 PRINT:PRINT:PRINT TAB(4); "EN CONDICIONES NORMALES, SA-"
425 PRINT "BIENDO QUE LA LONGITUD DEL BRA-"
428 PRINT "ZO ES ";L;" Y LA ACELERACION DEL"
430 PRINT "SISTEMA ES ";G;" EL PERIODO <T>"
435 PRINT "DEL PENDULO ES : "
438 PRINT:PRINT:PRINT
440 PRINT "*****"
450 PRINT " "
460 PRINT " ";P;" Segundos "; "*"
470 PRINT " "
480 PRINT "*****"
485 PRINT "(ESPERA UN MOMENTO, POR FAVOR)"
490 FOR I=1 TO 10000: NEXT I
495 GOTO 160
500 REM *****
502 REM * CALCULO DEL PERIODO <T> DEL PENDULO CUANDO ESTA *
503 REM * DENTRO DE UN SISTEMA CON UNA ACELERACION <a> *
504 REM *****
506 CLS
508 PRINT:PRINT
510 PRINT:PRINT:PRINT TAB(3); "EL PENDULO SE ENCUENTRA DENTRO"
515 PRINT "DE UN VEHICULO (coche, cohete,...)"
518 PRINT "EN MOVIMIENTO. EL SISTEMA SE VE"
520 PRINT "MODIFICADO DEBIDO A UNA ACELERA-"
522 PRINT "CION <a>."
530 PRINT: PRINT "POR TANTO: g.total = g + a "
535 PRINT: PRINT TAB(14); " GT = G + A "
536 PRINT:PRINT

```

```

538 PRINT:PRINT:PRINT: PRINT "ESPERA UN MOMENTO, POR FAVOR.1"
540 FOR I=1 TO 10000: NEXT I: CLS
544 REM *****
545 REM * INTRODUCCION DE LOS DATOS DE LA LONGITUD DEL BRAZO *
546 REM * Y DE LA ACELERACION DEL SISTEMA EN MOVIMIENTO *
547 REM *****
548 PRINT:PRINT
550 PRINT "LONGITUD DEL BRAZO DEL PENDULO"
552 PRINT "EN <METROS> : "
555 INPUT L
556 IF L=0 THEN CLS:PRINT "EL PERIODO <T> DEL PENDULO ES: T=0.": FOR I=1 TO 3000
: NEXT I:GOTO 160
558 PRINT:PRINT
559 PRINT:PRINT
560 PRINT:PRINT "ACELERACION QUE ALCANZA EL SISTEMA"
562 PRINT "EN MOVIMIENTO : "
563 INPUT A
565 IF A=0 THEN CLS:GOTO 320
570 LET GT=G+A
575 REM *** ALGORITMO PARA DETERMINAR EL PERIODO <T> ***
580 LET T=2*PI*(L/GT)^(1/2)
585 REM *** REDONDEO DEL VALOR DEL PERIODO <T> A DOS DECIMALES ***
590 LET P=INT(I*100)/100
595 REM *****
600 REM *****
605 REM * PRESENTACION DE LOS RESULTADOS EN PANTALLA *
607 REM *****
608 CLS
609 PRINT:PRINT:PRINT
610 PRINT:PRINT:PRINT TAB(3); "CUANDO EL PENDULO SE ENCUENTRA"
612 PRINT "DENTRO DE UN SISTEMA EN MOVIMIEN"
614 PRINT "TU, EL VALOR DEL PERIODO <T> DEL"
620 PRINT "PENDULO ES : "
625 PRINT:PRINT:PRINT
630 PRINT "*****"
640 PRINT " "
650 PRINT " " ; P ; " SEGUNDOS"; " "
660 PRINT " "
670 PRINT "*****"
675 PRINT:PRINT:PRINT:PRINT " (ESPERA UN MOMENTO, POR FAVOR) "
680 FOR I=1 TO 10000: NEXT I
690 GOTO 160

```

Modificaciones para otros equipos

Este programa es válido para el IBM-PC y compatibles, siendo necesarias las siguientes modificaciones para otros equipos:

ZX-SPECTRUM

```

150 IF U=1 THEN GOTO 300
154 IF U=2 THEN GOTO 500
156 GOTO 128
220 CLS: PRINT AT 12,10; "!!! FIN DE PROGRAMA !!!"
230 FOR I=1 TO 3000: NEXT I: CLS: GOTO 9999
319 IF L=0 THEN CLS: PRINT "PERIODO <T>: T= 0.":
FOR I=1 TO 1000: NEXT I: GOTO 160
540 FOR I=1 TO 5000: NEXT I: CLS
556 IF L=0 THEN CLS: PRINT "PERIODO <T>: T=0.":
FOR I=1 TO 1000: NEXT I: GOTO 160

```

COMMODORE

```

40 PRINT CHR$(147)
160 PRINT CHR$(147)
220 PRINT CHR$(147): PRINT "!!!FIN DE
PROGRAMA!!!"
230 FOR I=1 TO 5000: NEXT I: PRINT "PERIODO <T>:
T=0.": FOR I=1 TO 3000: NEXT I: GOTO 160
405 PRINT CHR$(147)
540 FOR I=1 TO 10000: NEXT I: PRINT CHR$(147)
556 IF L=0 THEN PRINT CHR$(147): PRINT "PERIODO"
<T>: T=0.": FOR I=1 TO 3000: NEXT I: GOTO 160
565 IF A=0 THEN PRINT CHR$(147): GOTO 320

```

AMSTRAD

```

220 CLS: LOCATE 10,12: PRINT " !!! FIN DE
PROGRAMA !!! "

```

MATEMATICAS

■ Recta en el plano afín. Formas de la ecuación de la recta

Un vector director de una recta es uno de los infinitos vectores libres que tienen su dirección.

Se pueden deducir diferentes formas de la ecuación de la recta conociendo un punto A (o dos puntos A y B) y un vector director de ella.

■ Formas de la ecuación de la recta

Componentes del punto A: A(A1,A2).

Componentes del punto B: B(B1,B2)

Componentes del vector director v:

$\vec{v}(V1,V2).$

Ecuación vectorial.

$$\overrightarrow{OX} = \overrightarrow{OA} + \alpha \cdot \vec{v} \quad [1]$$

Ecuaciones paramétricas

$$\begin{aligned} X &= A1 + \alpha \cdot V1 \\ Y &= A2 + \alpha \cdot V2 \end{aligned} \quad [2]$$

Ecuación continua. Se obtiene despejando el parámetro α de la paramétrica e igualando.

$$r \equiv \frac{X - A1}{V1} = \frac{Y - A2}{V2} \quad [3]$$

Cuando una de las componentes del vector director es nula, entonces la ecuación [3] toma forma simbólica al quedar uno de los dos cocientes dividido por 0.

Ecuación general de la recta. Se obtiene operando [3]:

$$\begin{aligned} (X - A1) \cdot V2 &= V1 \cdot (Y - A2) \\ V2 \cdot X - V2 \cdot A1 &= V1 \cdot Y - V1 \cdot A2 \\ V2 \cdot X - V2 \cdot A1 - V1 \cdot Y + V1 \cdot A2 &= 0 \\ V2 \cdot X - V1 \cdot Y + V1 \cdot A2 - V2 \cdot A1 &= 0 \end{aligned}$$

sea $A = V2$; $B = -V1$; y $C = V1 \cdot A2 - V2 \cdot A1$, de

donde

$$\text{ec. general. } Ax + By + C = 0 \quad [4]$$

Ecuación de la recta que pasa por dos puntos. Si tenemos dos puntos que pertenecen a la recta podemos deducir el valor del vector director y; con un punto y un vector director se obtienen todas las ecuaciones de la recta.

$$\begin{aligned} A(A1, A2) / \text{Un vector director } \vec{v} \text{ será:} \\ B(B1, B2) / \vec{v} (V1, V2) = (B1 - A1, B2 - A2) \end{aligned}$$

Ecuación en forma explícita de la recta.

Es de la forma $r \equiv M \cdot x + N$.

$$\begin{aligned} r \equiv M \cdot x + N, \text{ donde } M &= V2/V1, \text{ y} \\ N &= V1 \cdot A2 - V2 \cdot A1 \\ (A1, A2): \vec{v}(V1, V2) \end{aligned}$$

Ecuación punto pendiente de la recta. Se obtiene cuando se conoce un punto que pertenece a la recta y su pendiente $< M >$.

$$\begin{aligned} r \equiv y - A2 &= M \cdot (x - A1) \\ M &= V2/V1 \\ A(A1, A2) \end{aligned}$$

```

10 REM *****
15 REM *
20 REM *          PLANO AFIN          *
25 REM *
30 REM *  ECUACIONES DE LA RECTA  *
35 REM *
40 REM *****
45 CLS
50 PRINT:PRINT:PRINT TAB(4)"PUEDES DETERMINAR LA ECUACION"
55 PRINT "DE LA RECTA EN SUS FORMAS MAS CO"
60 PRINT "NOCIDAS SEGUN LAS CONDICIONES"
65 PRINT "SIGUIENTES:"
70 PRINT:PRINT "1.- DADOS UN PUNTO <A> Y UN VEC"
75 PRINT TAB(5);"TOR DIRECTOR <V> DE LA RECTA"
80 PRINT:PRINT "2.- CONOCIDOS DOS PUNTOS A Y B"
85 PRINT TAB(5);"QUE PERTENECEN A LA RECTA"
90 PRINT:PRINT "3.- CONOCIENDO UN PUNTO <A> Y LA"
95 PRINT TAB(5);"PENDIENTE <M> DE LA RECTA"
100 PRINT:PRINT "4.- CONOCIENDO LOS PUNTOS DE CO"
105 PRINT TAB(5);"ORDENADAS"
110 PRINT:PRINT:PRINT "PULSA LA OPCION DESEADA (1-2-3-4): "
120 INPUT U
130 CLS
140 ON U GOTO 300,400,500,600
150 CLS
160 PRINT:PRINT:PRINT:PRINT "QUIERES DETERMINAR LA ECUACION"
165 PRINT "DE OTRA RECTA (S/N) ";
168 INPUT K$
170 IF K$="S" OR K$="s" THEN GOTO 45
180 IF K$(">"N" AND K$(">"n" THEN GOTO 160
190 CLS: LOCATE 12,10: PRINT "!!! ADIOS !!!"
195 FOR I=1 TO 5000:NEXT I :CLS
200 END
300 REM *****
301 REM *  SUBROUTINA PARA INTRODUCIR LAS COORDENADAS DE  *
302 REM *  UN PUNTO Y UN VECTOR DIRECTOR DE LA RECTA  *
303 REM *****
305 CLS
308 PRINT:PRINT
310 PRINT:PRINT: INPUT "COORDENADAS DEL PUNTO A(A1,A2): ";A1,A2
320 PRINT:PRINT: INPUT "COORDENADAS DEL VECTOR DIRECTOR V(V1,V2): ";V1,V2
330 GOSUB 900

```

APRENDER CON EL ORDENADOR

```

340 GOTO 150
400 REM *****
401 REM * SUBROUTINA PARA INTRODUCIR LAS COORDENADAS DE *
402 REM * DOS PUNTOS QUE PERTENECEN A LA RECTA *
403 REM *****
405 CLS
408 PRINT:PRINT
410 PRINT:PRINT: INPUT "COORDENADAS DEL PUNTO A(A1,A2): ";A1,A2
420 PRINT:PRINT: INPUT "COORDENADAS DEL PUNTO B(B1,B2): ";B1,B2
430 LET V1=B1-A1
440 LET V2=B2-A2
450 GOSUB 900
460 GOTO 150
500 REM *****
501 REM * SUBROUTINA PARA INTRODUCIR LAS COORDENADAS DE UN PUNTO *
502 REM * Y EL VALOR DE LA PENDIENTE <M> DE LA RECTA *
503 REM *****
505 CLS
508 PRINT:PRINT
510 PRINT:PRINT: INPUT "COORDENADAS DEL PUNTO A(A1,A2): ";A1,A2
520 PRINT:PRINT: INPUT "VALOR DE LA PENDIENTE <M> DE LA RECTA: ";M
530 LET V1=1:LET V2=M
540 GOSUB 900
550 GOTO 150
600 REM *****
601 REM * SUBROUTINA PARA INTRODUCIR LOS VALORES DE LOS PUNTOS *
602 REM * DE CORTE CON LOS EJES <X> E <Y> *
603 REM *****
605 CLS
608 PRINT:PRINT
610 PRINT:PRINT: INPUT "PUNTO DONDE CORTA AL EJE <X> CON SU SIGNO: ";X
620 PRINT:PRINT: INPUT "PUNTO DONDE CORTA AL EJE <Y> CON SU SIGNO: ";Y
630 LET A1=X: LET A2=0
640 LET V1=X: LET V2=-Y
650 GOSUB 900
660 GOTO 150
900 REM *****
901 REM * PRESENTACION EN PANTALLA DE LOS RESULTADOS *
902 REM *****
905 CLS
910 IF V1=0 AND V2=0 THEN GOTO 1130
930 PRINT "ECUACIONES PARAMETRICAS:"
940 PRINT: PRINT "X = ";A1;" +  $\alpha$  * (";V1;" )
950 PRINT "Y = ";A2;" +  $\alpha$  * (";V2;" )
960 PRINT
970 PRINT "ECUACION EN FORMA CONTINUA:"
980 PRINT: PRINT "X - ";A1;" Y - ";A2
990 PRINT "----- = -----"
1000 PRINT " ";V1;" ";V2
1010 PRINT
1020 PRINT "ECUACION GENERAL (IMPLICITA):"
1030 PRINT:PRINT "V2;" X + ";-V1;" Y + ";V1*A2-V2*A1;" = 0"
1040 PRINT
1050 PRINT "ECUACION EN FORMA EXPLICITA:"
1055 PRINT
1060 IF V1=0 THEN PRINT "Y = ";A1: GOTO 1080
1070 PRINT "Y = ";V2/V1;" X + ";(A2*V1-V2*A1)/V1
1080 PRINT
1090 PRINT "ECUACION PUNTO-PENDIENTE:"
1100 PRINT: IF V1=0 THEN PRINT "Y = ";A1: GOTO 1150
1110 PRINT "Y - ";A2;" = ";V2/V1;" (X - ";A1;" )":GOTO 1150
1130 PRINT:PRINT:PRINT "NO HAY ECUACION DE LA RECTA PUES"
1135 PRINT "EL VECTOR DIRECTOR ES EL VECTOR"
1140 PRINT "NULO: V(0,0).":FOR I=1 TO 5000:NEXT I:GOTO 150
1150 FOR I=1 TO 20000: NEXT I
1160 RETURN

```

Modificaciones para otros equipos

Son necesarias las siguientes modificaciones para otros equipos:

ZX-SPECTRUM

```

140 IF U=1 THEN GOTO 300
142 IF U=2 THEN GOTO 400
144 IF U=3 THEN GOTO 500
146 IF U=4 THEN GOTO 600

```

```

148 GOTO 110
190 CLS: PRINT AT 12,10;" ¡¡¡ ADIOS !!! "
195 FOR I=1 TO 2000: NEXT I: GOTO 9999
1130 FOR I=1 TO 10000: NEXT I

```

COMMODORE

```

45 PRINT CHR$(147)
130 PRINT CHR$(147)
150 PRINT CHR$(147)
190 PRINT CHR$(147): PRINT " ¡¡¡ ADIOS !!! "

```

AMSTRAD

190 LOCATE 10,12: PRINT "¡¡¡ADIOS!!!"

SOCIEDAD

■ La pintura española durante el siglo XIX

Durante el primer tercio del siglo XIX los pintores españoles, excepto los seguidores de Goya, quedaron sometidos a la influencia del Neoclasicismo. Entre éstos destacan el valenciano Vicente López, retratista de la familia real y de los principales personajes políticos de la época; y el santanderino José de Madrazo, muy influenciado por el pintor francés David, conocido por sus cuadros históricos.

En el segundo tercio del siglo la pintura romántica cuenta con tres importantes figuras cultivadoras del retrato: Federico de Madrazo (el retratista más destacado de la España romántica), el sevillano Antonio María Esquivel (con el famoso cuadro *Los Poetas*), y el catalán Joaquín Espalter, a quien pertenece el cuadro *La familia Flaquer*, del Museo Romántico de Madrid.

Al iniciarse el último tercio del siglo adquirió un gran desarrollo la pintura de cuadros de historia, género puesto de moda en Francia por David y sus seguidores. Los cuadros de historia se caracterizan por su gran tamaño y su extraordinaria minuciosidad, interesando en ellos más la precisión histórica que la calidad técnica.

Entre los muchos pintores que cultivaron este género destacan principalmente José Casado de Alisal, cuya obra más representativa es la *Rendición de Bailén*; Antonio Gisbert, que trata temas relacionados con la libertad

como el *Fusilamiento de Torrijos y sus compañeros*; y el más importante de todos, Eduardo Rosales, con su célebre cuadro el *Testamento de Isabel la Católica*, que triunfó en la Exposición de 1864.

Contemporáneo de Rosales es el pintor catalán Mariano Fortuny. La luz y el color, que le fascinaban, los encontró en Marruecos, adonde se trasladó con ocasión de la guerra hispano-marroquí en tiempos de O'Donnell, pintando entre otras la *Batalla de Tetuán*. Renunció a la pintura de la historia, prefiriendo dedicarse a temas sencillos e intrascendentes ejecutados con una técnica minuciosa y dotados de intenso colorido. Su obra más importante de este género es *La Vicaría*.

El impresionismo español cuenta con un buen número de pintores que cultivaron principalmente el paisaje. Entre ellos sobresalen: Aureliano Beruete, Darío de Regoyos, Eliseo Meifrén, Joaquín Mir y Santiago Rusiñol. Pero las dos figuras más importantes de la última etapa del siglo XIX y primeros decenios del XX son: el valenciano Joaquín Sorolla y el vasco Ignacio Zuloaga.

Sorolla, el pintor de Valencia, comenzó pintando cuadros de historia como el *Dos de Mayo*, a los que siguieron después cuadros de tema social como *¡Aún dicen que el pescado es caro!* Su verdadera innovación fue la pintura realizada al aire libre en las luminosas playas valencianas, pintura llena de luz y de colorido y ejecutada con una técnica muy suelta.

Ignacio Zuloaga es el pintor de Castilla, especialmente de las tierras de Avila, y sobre todo de Segovia, en donde vivió muchos años. Excelente paisajista, no dudó en colocar amplios fondos de paisaje en sus retratos. Una de sus obras más conocidas es el *Cristo de la sangre*, que tiene la ciudad de Avila al fondo.

```
10 REM *****
15 REM * CONOCIMIENTOS SOBRE PINTURA *
20 REM *   ESPAÑOLA DEL SIGLO XIX   *
25 REM *****
30 CLS
40 LET A=0
50 RANDOMIZE TIMER
60 DIM A$(10): DIM B$(40): DIM N(10)
70 REM *****
75 REM *   LECTURA DE PREGUNTAS   *
80 REM *****
90 FOR I=1 TO 10
100 READ A$(I)
110 NEXT I
120 REM *****
125 REM *   LECTURA DE RESPUESTAS   *
130 REM *****
```

APRENDER CON EL ORDENADOR

```

140 FOR J=1 TO 40
150 READ B$(J)
160 NEXT J
170 REM *****
175 REM * LECTURA DE SOLUCIONES *
180 REM * A LAS RESPUESTAS *
185 REM *****
190 FOR K=1 TO 10
195 READ N(K)
200 NEXT K
210 REM *****
215 REM * EDICION EN PANTALLA DE *
220 REM * PREGUNTAS Y RESPUESTAS *
225 REM *****
230 FOR I=1 TO 10
240 PRINT:PRINT TAB(2);A$(I)
250 PRINT:PRINT
260 PRINT:PRINT TAB(6);B$(I*4-3)
270 PRINT:PRINT TAB(6);B$(I*4-2)
280 PRINT:PRINT TAB(6);B$(I*4-1)
290 PRINT:PRINT TAB(6);B$(I*4)
300 PRINT:PRINT
310 REM *****
315 REM * INTRODUCCION DE LA *
320 REM * SOLUCION CORRECTA *
325 REM *****
330 INPUT "LA RESPUESTA ES: ";R
340 IF R=N(I) THEN GOSUB 500
350 IF R<>N(I) THEN GOSUB 800
360 NEXT I
370 PRINT:PRINT:PRINT
380 REM *****
385 REM * EDICION DEL FIN DEL *
390 REM * PROGRAMA *
395 REM *****
400 PRINT TAB(3);"RESPUESTAS CORRECTAS: ";A
410 PRINT:PRINT
415 IF A<1 OR A>10 THEN GOTO 400
420 IF A>=1 AND A<4 THEN PRINT "MUY DEFICIENTE":GOTO 490
430 IF A=4 THEN PRINT "INSUFICIENTE":GOTO 490
440 IF A=5 THEN PRINT "SUFICIENTE":GOTO 490
450 IF A=6 THEN PRINT "BIEN":GOTO 490
460 IF A>6 AND A<9 THEN PRINT "NOTABLE":GOTO 490
470 IF A=9 THEN PRINT "SOBRESALIENTE":GOTO 490
480 PRINT "MUY BIEN. PERFECTO. TIENES UN 10 DE NOTA"
490 FOR I=1 TO 2000:NEXT I:CLS:END
500 REM *****
505 REM * SUBROUTINA DE RESPUESTAS *
510 REM * CORRECTAS *
515 REM *****
520 CLS
530 PRINT:PRINT:PRINT
540 PRINT "¡SOLUCION CORRECTA!"
550 LET A=A+1
560 FOR F=0 TO 2000:NEXT F:CLS
570 RETURN
800 REM *****
805 REM * SUBROUTINA DE RESPUESTAS *
810 REM * INCORRECTAS *
815 REM *****
820 CLS
830 PRINT:PRINT:PRINT
840 PRINT "¡NO ES CORRECTO!"
850 PRINT:PRINT
860 PRINT "LA SOLUCION CORRECTA ES: "
865 PRINT:PRINT B$(I*4+N(I)-4)
870 FOR F=1 TO 5000: NEXT F:CLS
880 RETURN
1000 REM *****
1010 REM * DATAS CON LAS PREGUNTAS *
1020 REM * RESPUESTAS Y CON LAS *
1030 REM * SOLUCIONES CORRECTAS *
1040 REM *****
1050 DATA "OBRA CONOCIDA DE EDUARDO ROSALES :", "QUE RECREO ZULOAGA EN SUS CUADI
OS: ", "QUIEN INFLUENCIO A LOS PINTORES DEL ULTIMO TERCIO DE SIGLO: ", "REPRESEN
ANTES DEL IMPRESIONISMO ESPAÑOL: "
1060 DATA "PINTORES DEL SEGUNDO TERCIO DE SIGLO CULTIVADORES DEL RETRATO:", "AUT
R DE LA OBRA <FUSILAMIENTO DE TORRIJOS>:", "QUE BUSCABA FORTUNY RECREAR EN SUS CU
DROS:"
1070 DATA "PINTORES MAS DESTACADOS DE ULTIMOS DEL S. XIX Y PRINCIPIOS DEL S. XX
", "SOROLLA PLASMO CON GRAN REALISMO:"

```

```

1080 DATA "QUE INFLUENCIA SUFRIERON LOS PINTORES DEL PRIMER TERCIO DEL SIGLO (E.
CEPTO LOS SEGUIDORES DE GOYA):"
1090 DATA "1. RENDICION DE BAILEN","2. LA VICARIA","3. TESTAMENTO DE ISABEL LA
ADOLICA", "4. LA FAMILIA FLAQUER"
1100 DATA "1. LOS PAISAJES CASTELLANOS","2. HECHOS HISTORICOS RELEVANTES","3. LI
S AMBIENTES POPULARES","4. EL MOVIMIENTO DE MASAS"
1110 DATA "1. DELACROIX","2. DAVID Y SEGUIDORES","3. T. ROUSSEAU", "4. MANET"
1120 DATA "1. ZULOAGA Y GISBERT","2. MEIFREN Y GISBERT","3. REGOYOS Y ROSALES",
4. RUSIÑOL Y BERUETE"
1130 DATA "1. FORTUNY Y SOROLLA","2. F. MADRAZO Y ESPARTEL","3. F. MADRAZO Y REI
OYOS", "4. F. MADRAZO Y J. MADRAZO"
1140 DATA "1. GISBERT","2. CASADO DEL ALISAL","3. ROSALES","4. FORTUNY"
1150 DATA "1. LUZ Y EL COLOR","2. PRECISION HISTORICA","3. MINUCIOSIDAD EN EL D
BUJO","4. EL PAISAJE CASTELLANO"
1160 DATA "1. SOROYA Y ROSALES", "2. ZULOAGA Y RUSIÑOL","3. SOROLLA Y MADRAZO",
4. ZULOAGA Y SOROLLA"
1170 DATA "1. LOS AMBIENTES POPULARES","2. COLORIDO DE LAS PLAYAS VALENCIANAS",
3. LOS PAISAJES","4. EL DETALLE HISTORICO"
1180 DATA "1. TENEBRISMO ITALIANO","2. IMPRESIONISMO FRANCES","3. NEOCLASICISMO
","4. REALISMO"
1190 DATA 3,1,2,4,2,1,1,4,2,3

```

Modificaciones para otros equipos

Son necesarias las modificaciones siguientes:

ZX-SPECTRUM

```

50 (NO ES NECESARIA)
60 DIM A$(10,40): DIM B$(40,30): DIM N(10)
490 FOR I=1 TO 1000: NEXT I: CLS: GOTO 9999
500 FOR F=1 TO 1000: NEXT F: CLS
870 FOR F=1 TO 3000: NEXT F: CLS

```

COMMODORE

```

30 PRINT CHR$(147)
490 FOR I=1 TO 2000: NEXT I: PRINT CHR$(147): END
500 FOR F=1 TO 2000: NEXT F: PRINT CHR$(147)
870 FOR F=1 TO 5000: NEXT F: PRINT CHR$(147)

```

PARA LOS MAS JOVENES

■ Sistemas de ecuaciones lineales de dos ecuaciones con dos incógnitas

Un sistema de ecuaciones lineales de dos ecuaciones con dos incógnitas puede considerarse del tipo:

$$\begin{aligned} Ax + By &= C \\ Dx + Ey &= F \end{aligned}$$

Para resolver este tipo de sistema es necesario llegar a determinar el par (x,y) común que satisface ambas ecuaciones, o lo que

es lo mismo, determinar los números para los cuales se cumplen simultáneamente las dos ecuaciones del sistema. Para ello se van simplificando las ecuaciones de tal modo que se consigan ecuaciones lo más sencillas posibles hasta poder despejar las incógnitas.

■ Métodos de resolución

Existen varios métodos de resolución de sistemas de ecuaciones lineales de dos ecuaciones con dos incógnitas:

1. *Método de sustitución.* Se despejará una de las incógnitas de una de las ecuaciones y se sustituye en la otra.

2. *Método de igualación.* Despejamos de ambas ecuaciones la misma incógnita y se igualan las expresiones que resultan.

3. *Método de reducción.* Se multiplica cada una de las dos ecuaciones por un número distinto de cero, de forma que posteriormente al sumar o restar ambas ecuaciones desaparezca una de las incógnitas.

El programa que se plantea a continuación sirve para que puedas resolver el sistema de dos ecuaciones con dos incógnitas donde los coeficientes pueden tomar cualquier valor que tú introduzcas. El ordenador te dará, posteriormente, el valor de las incógnitas X e Y que satisfacen ambas ecuaciones.

```

10 REM *****
20 REM *
22 REM * SISTEMA DE ECUACIONES LINEALES DE DOS *
25 REM * ECUACIONES CON DOS INCOGNITAS *
30 REM *
35 REM *****
40 PRINT: PRINT TAB(10);"SISTEMA DE ECUACIONES"
45 PRINT TAB(10);"-----"

```

```

50 PRINT: PRINT TAB(5); "UN SISTEMA DE ECUACIONES LI"
55 PRINT: PRINT "NEALES DE ECUACIONES CON DOS INCOG"
60 PRINT: PRINT "NITAS PUEDE CONSIDERARSE DEL TIPO:"
70 PRINT: PRINT TAB(10); " A.x + B.y = C"
80 PRINT: PRINT TAB(10); " D.x + E.y = F"
90 FOR I=1 TO 2000: NEXT I
100 REM ***** INTRODUCCION DE LOS COEFICIENTES *****
110 PRINT: PRINT: PRINT TAB(3); "INTRODUCE EL VALOR DE CADA COE"
112 PRINT: PRINT "FICIENTE A MEDIDA QUE SE TE PIDA"
115 FOR I=1 TO 3000: NEXT I: CLS
118 PRINT: PRINT: PRINT: PRINT
120 PRINT: INPUT "1a. ECUACION. COEFICIENTE <A> : "; A
130 PRINT: INPUT "1a. ECUACION. COEFICIENTE <B> : "; B
135 IF A=0 AND B=0 THEN CLS: PRINT: PRINT: PRINT: PRINT "COEFICIENTES <A> Y <B> NO V
ALIDOS.": FOR I=1 TO 1000: NEXT I: CLS: GOTO 110
140 PRINT: INPUT "1a. ECUACION. COEFICIENTE <C> : "; C
150 PRINT: INPUT "2a. ECUACION. COEFICIENTE <D> : "; D
160 PRINT: INPUT "2a. ECUACION. COEFICIENTE <E> : "; E
165 IF D=0 AND E=0 THEN CLS: PRINT: PRINT: PRINT: PRINT "COEFICIENTES <D> Y <E> NO
VALIDOS.": FOR I=1 TO 1000: NEXT I: CLS: GOTO 110
170 PRINT: INPUT "2a. ECUACION. COEFICIENTE <F> : "; F
180 CLS
190 REM **** POSIBLE CORRECCION DE DATOS *****
200 PRINT: PRINT: PRINT "EL SISTEMA A RESOLVER ES: "
210 PRINT: PRINT: PRINT A; " x + "; B; " y = "; C
220 PRINT: PRINT D; " x + "; E; " y = "; F
230 PRINT: PRINT: PRINT: PRINT "¿TODOS LOS DATOS SON CORRECTOS (S/N)? "
240 PRINT: INPUT M$
250 IF M$="S" OR M$="s" THEN GOTO 300
260 IF M$(">"N" AND M$("<"n" THEN GOTO 230
270 PRINT: PRINT: PRINT "INTRODUCE LOS DATOS CORRECTAMENTE OTRA VEZ": FOR I=1 TO 2
000: NEXT I: CLS: GOTO 120
300 REM *****
302 REM * RESOLUCION DEL SISTEMA DE DOS ECUACIONES *
304 REM *****
305 CLS
310 LET Y = (A*F-D*C)/(A*E-D*B)
320 LET X = (C-B*Y)/A
330 REM *****
340 REM * PRESENTACION EN PANTALLA DE LOS RESULTADOS *
350 REM *****
360 PRINT "FORMA GENERAL DEL SISTEMA DE DOS ECUACIONES: "
370 PRINT: PRINT TAB(7); "A.x + B.y = C"
380 PRINT: PRINT TAB(7); "C.x + D.y = F"
390 PRINT: PRINT: PRINT " EL SISTEMA A RESOLVER CON LOS COEFICIENTES DADOS ES: "
400 PRINT: PRINT TAB(7); A; " x + "; B; " y = "; C
410 PRINT: PRINT TAB(7); D; " x + "; E; " y = "; F
420 PRINT: PRINT: PRINT " EL PAR (X,Y) QUE SATISFACE LAS DOS ECUACIONES ES:"
430 PRINT: PRINT TAB(7); " x = "; X
440 PRINT: PRINT TAB(7); " y = "; Y
450 FOR I=1 TO 10000: NEXT I
455 REM ***** POSIBILIDAD DE CONTINUAR O SALIR DEL PROGRAMA *****
460 PRINT: INPUT "DESEAS RESOLVER OTRO SISTEMA (S/N) "; V$
470 IF V$="S" OR V$="s" THEN CLS: GOTO 50
480 IF V$(">"N" AND V$("<"n" THEN GOTO 460
490 CLS: LOCATE 12,10: PRINT " !!! FIN DEL PROGRAMA !!! "
500 FOR I=1 TO 2000: NEXT I: CLS: END

```

Modificaciones para otros equipos

Son necesarias las modificaciones siguientes:

ZX-SPECTRUM

```

90 FOR I=1 TO 1000:NEXT I
490 CLS: PRINT AT 12,10; " !!! FIN DE PROGRAMA !!! "
500 FOR I=1 TO 1000:NEXT I: CLS: GOTO 9999

```

COMMODORE

```

38 PRINT CHR$(147)
115 FOR I=1 TO 3000: NEXT I: PRINT CHR$(147)
135 IF A=0 AND B=0 THEN PRINT CHR$(147): PRINT:
PRINT: PRINT "COEFICIENTES <A> Y <B> NO
VALIDOS": FOR I=1 TO 1000: NEXT I: PRINT CHR$(147):
GOTO 110

```

```

165 IF D=0 AND E=0 THEN PRINT CHR$(147): PRINT:
PRINT: PRINT "COEFICIENTES <D> Y <E> NO
VALIDOS": FOR I=1 TO 1000: NEXT I: PRINT CHR$(147):
GOTO 110
180 PRINT CHR$(147)
270 PRINT: PRINT: PRINT "INTRODUCE LOS DATOS
CORRECTAMENTE OTRA VEZ": FOR I=1 TO 2000: NEXT
I: PRINT CHR$(147): GOTO 120
470 IF V$="S" OR V$="s" THEN PRINT CHR$(147): GOTO
50
490 PRINT CHR$(147): PRINT " !!! FIN DE PROGRAMA !!!
"
500 FOR I=1 TO 2000: NEXT I: PRINT CHR$(147): END

```

AMSTRAD

```

490 CLS: LOCATE 10,12: PRINT " !!! FIN DE PROGRAMA
!!! "

```

PEQUEÑA HISTORIA DE LA INFORMATICA

LAS CALCULADORAS DESDE EL SIGLO XIX HASTA NUESTROS DIAS



A dijimos anteriormente que con Babbage prácticamente murió la idea de máquina universal, que pudiera ocuparse de cualquier cómputo automáticamente, siendo además capaz de llevar a cabo algunas opera-

ciones lógicas.

Este hecho, sin embargo, no impidió que las simples máquinas de calcular, sin aspiraciones tan elevadas como la máquina de Babbage, sufrieran modificaciones muy importantes. La máquina de Pascal y Leibniz siguieron mejorándose y, además, apareció un mercado floreciente, que iba creciendo día a día, casi hora por hora: el mercado del servicio a las empresas.

La industrialización supuso un avance enorme para la humanidad (desde el punto de vista técnico), aunque en el plano humano fuera la causa inmediata de tanta miseria. Pero no tenemos intención de escribir *David Copperfield* (somos mucho más modestos). Sin embargo, con el empuje de las nuevas industrias, las necesidades de simplificar la labor de oficina se iban imponiendo, y aunque la mano de obra fuera muy barata, la fiabilidad de las máquinas y su rapidez iban atrayendo cada vez más empresarios que se decidían a comprarlas.

Uno de los aspectos que mejoró sustancialmente fue la presentación de los resultados. Estos se daban impresos en papel.

Las multiplicaciones y divisiones se obtenían desde las primeras máquinas de calcular por sumas reiterativas. Este sistema era una lacra para alcanzar una velocidad de cálculo aceptable, y además suponía multitud de problemas de arrastres, etc. En 1889 León Bollée, con sólo diecinueve años, inventa un nuevo sistema de multiplicar muy ingenioso, que

multiplica directamente, con una tabla preconfigurada mediante una matriz de agujas. El sistema era muy práctico, y Leon prosiguió intentando mecanizar la división directa. Sin embargo, no llegó a resolver el problema, y lo abandonó para interesarse por los trabajos de su padre.

Es interesante saber que Leon Bollée era uno de los miembros de una familia de inventores. Su padre, Amadeo Bollée, construyó un coche de vapor que hizo el recorrido París-Le Mans en 1873. Pero ahí no acabó todo. Una vez abandonado el motor a vapor, en los años siguientes a la Primera Guerra Mundial, Amadeo (hijo) y su hermano León se hicieron famosos mejorando los motores de explosión. Además, fueron los fundadores del circuito de Le Mans.

El prototipo de la máquina de Bollée se quedó en poco más que eso, un prototipo, ya que sólo se fabricaron tres modelos y apenas si se vendieron. Sin embargo, cinco años más tarde O. Steiger creó una nueva máquina, basada esencialmente en la de Bollée, y se hizo rico. En sólo tres años vendió más de mil máquinas. El nombre de la máquina era muy gracioso, se llamaba el «Millonario», y ha sido probablemente la máquina de calcular más popular de todos los tiempos. Sin embargo, esta máquina, aunque multiplicaba directamente, no disponía de división directa. Copias y modificaciones de este mismo sistema tuvieron el mismo éxito comercial, y Egli, de Zurich, vendió más de 5.000 máquinas (eso sí, en cuarenta años).

El «Millonario» fue pensado por su creador para que resolviera fundamentalmente problemas comerciales, cambios de moneda, tipos de interés, etc. Sin embargo, pasados unos años, se utilizó también mucho para labores científicas. Como siempre, la Astronomía

y la Meteorología, ciencias que manejan muchos datos, fueron las disciplinas que más máquinas de este tipo absorbieron. El descubrimiento del planeta Plutón se basó en múltiples cálculos e investigaciones, asistidas por la pequeña máquina tan popular en su época.

La división directa se consigue al otro lado del mar, en los Estados Unidos. El diseño es de J. R. Monroe, que, como indicamos anteriormente, creó su propia compañía, que prosperó enormemente, existiendo en nuestros días.

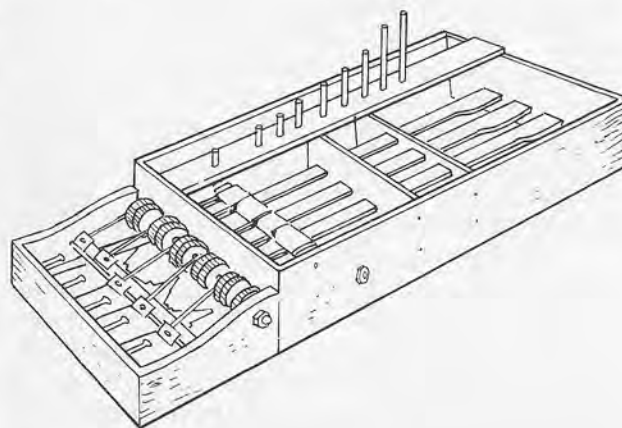
Sin embargo, nosotros consideramos que el avance mayor corresponde directamente a la presentación de los resultados. En las primeras máquinas éstos se leían directamente de pequeñas ventanas, y sólo había posibilidad de mantenerlos sin que se perdiesen escribiéndolos a mano. Sabemos que Babbage había diseñado un sistema de almacenamiento de resultados, y otro de presentación impresa, pero recordamos al lector que las miras de Babbage eran muchísimo más altas que las de los pequeños fabricantes de calculadoras, desde la propia raíz del problema. Esa fue quizá la razón de que nunca llegaran a llevar a la práctica sus ideas. Sin embargo, las calculadoras comenzaron a sacar resultados impresos hacia 1875, aunque realmente la primera máquina que apareció con la posibilidad de presentar resultados en papel no fue una simple calculadora, era realmente una máquina registradora (una caja para el dinero capaz de realizar las operaciones elementales). Su creador fue un francés, Pottin, pero no está muy claro si su diseño era original o una copia de algún otro diseño anterior, modificado.

El mercado de las máquinas de calcular cambió sustancialmente. Ya no eran máquinas para realizar trabajos aburridos y mecánicos, ni científicos. Se trataba de máquinas para acelerar y activar las operaciones comerciales. Y naturalmente, la enseña pasó a un país que estaba creciendo mucho más rápidamente que la vieja Europa, los Estados Unidos de Norteamérica. En este joven país, Felt, un técnico muy pragmático, creó una máquina a la que llamó «contómetro». Tuvo éxito, ya que había estudiado muy cuidadosamente las necesidades de los usuarios, procurando adaptarse y satisfacerles en lo posible. Poco tiempo después, Bourroughs creó una máquina muy completa, e intentó comercializarla desde el negocio de su padre, la American Arithmometer Company. (Bourroughs era muy aficionado a la mecánica, su padre era mecánico, y él ha-

bía nacido y se había criado en ese ambiente.) Sin embargo, la máquina que diseñó era tan complicada que prácticamente había que llamarle cada una de las veces para que explicara a cada usuario cómo hacer funcionar su máquina. El intento fue un fracaso, pero Bourroughs no se desanimó. Tenía fe en sí mismo, y no era soberbio. Aceptó las críticas. Sin embargo, podemos imaginar el mal humor que tendría cuando, aceptando su fracaso, y que las máquinas eran invendibles, se dirigió al almacén de su padre y, sin mediar palabra, tiró por la ventana unas cincuenta máquinas que no se habían vendido hasta el momento. Bravo chico. El asunto no quedó ahí, y Bourroughs se quitó la espina, diseñando una máquina de manejo mucho más sencillo que tuvo un enorme éxito. Su compañía (con otro nombre, para nosotros casi tan complicado como el que eligió su padre) permanece hasta nuestros días, y es uno de los fabricantes de ordenadores más importantes. La máquina se llamaba curiosamente «Máquina de sumar y listar».

Bourroughs puede anotar en su haber otra modificación importante, que sigue utilizándose en nuestros días (aunque mejorada). En los últimos modelos de registradoras Bourroughs, éste había incorporado un sistema de grabación sobre papel de los datos de entrada. Esta innovación puede parecer trivial, pero no lo es en absoluto, ya que de este modo se pueden solventar rápidamente posibles errores, quedando además constancia de todas las operaciones completas, para posteriores manipulaciones o verificaciones.

Los progresos de las máquinas de calcular fueron creciendo y las máquinas mejo-



«Caja de macarrones».

rando paulatinamente, hasta aproximadamente la Primera Guerra Mundial. A partir de este momento, ya conseguidos bastantes objetivos, los puntos de mira comienzan a cambiar muy lentamente. Sin embargo, estas máquinas sirvieron para tener resueltos muchos pequeños problemas que surgieron en los primeros ordenadores, y, por tanto, ayudaron considerablemente a su desarrollo. Uno de estos pequeños problemas que quedaron resueltos fue la introducción de órdenes y datos mediante tarjetas perforadas, y otro la presentación de resultados impresos, como constancia escrita de la información obtenida. Además, el mercado se abrió, y era propicio a nuevas máquinas más potentes y eficaces que las sencillas calculadoras-registradoras.

En cualquier caso, y a pesar de los enormes avances, apenas si se hubiera llegado mucho más lejos. Los engranajes son un sistema muy burdo de tratar la información. Con el dominio de la electricidad, y más tarde de la electrónica, se dio el salto trascendental para llegar a los ordenadores de hoy. La energía eléctrica es mucho más fácilmente cuantificable que cualquier otro tipo de energía.

La primera caja registradora sabemos que es americana. Su creador, James Ritty, la diseñó «adaptando» a las nuevas necesidades un contador de las revoluciones de la hélice de un barco. La máquina realizaba las operaciones básicas, pero no sumaba totales, creando subtotales parciales. Esta operación se incluyó en modelos posteriores, a los que se adaptó también una caja para guardar el dinero, y una pequeña campanilla o timbre que sonaba cuando se abría el cajón del dinero. En realidad, la empresa de Ritty puede considerarse el embrión del que surgió la que luego sería National Cash Register (NCR). En 1882 Ritty vendió su compañía a Jacob Eckert (en otro momento hablaremos de otro Eckert, personaje muy importante en la historia de la Informática). Finalmente, John Patterson compró la compañía a Eckert, cambiándole el nombre a National Cash Register (NCR), compañía que aún subsiste, siendo uno de las más importantes fabricantes de ordenadores.

No debemos olvidarnos de uno de los primeros españoles, que aparecen en nuestra historia, desde el medieval Raimundo Lulio. Ramón Vereá diseñó una pequeña máquina que realizaba las operaciones elementales. Su éxito no fue demasiado grande, y sobre todo poco después apareció el «Millonario», que arrolló, absorbiendo la mayor parte del mer-

¿Sabía usted que...

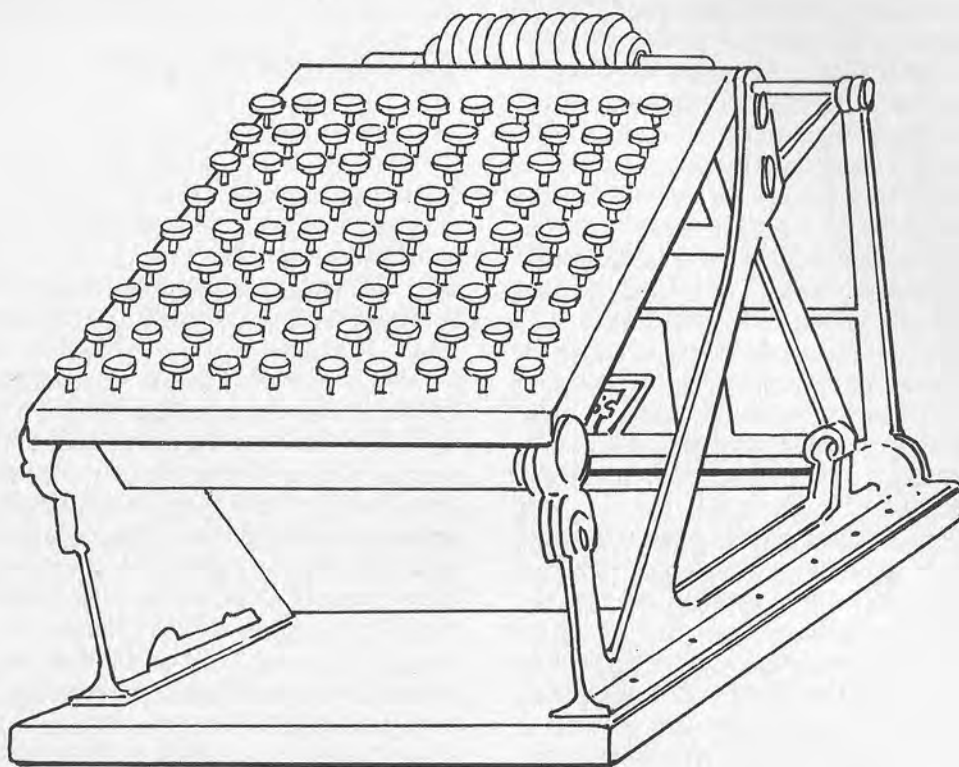
¿Se ha parado usted a pensar en que, debido al uso indiscriminado de las calculadoras, los niños actuales pueden olvidar cómo se suma o se resta, y al mismo tiempo qué están haciendo cuando suman o restan una cantidad?

Es cierto, con frecuencia, al calcular el precio de un artículo rebajado un diez o un veinticinco por ciento, el empleado saca su máquina de calcular para obtener la cifra exacta. Sin embargo, el cálculo mental es importante, es una disciplina útil para saber «dónde se encuentra uno» o detectar posibles errores en la manipulación de la calculadora. No todo el mundo tiene la misma capacidad para calcular mentalmente, aunque es una cualidad, como muchas otras, educable. Una de las personas que se han hecho famosas por este tipo de habilidad es el señor William Klein, del Centro Europeo para la Investigación Nuclear (CERN), de Ginebra. Este señor era capaz de evaluar fórmulas matemáticas complejas, siendo utilizado en el centro para ayudar a analistas y programadores a optimizar los programas. El señor Klein era capaz de realizar mentalmente multiplicaciones de cuatro cifras con una rapidez increíble. Conoce los logaritmos de los números hasta 150 con catorce decimales, y puede realizar muchos otros cálculos, como, por ejemplo, qué día de la semana fue determinada fecha histórica.

Aunque era hijo de médico, y comenzó a su vez la carrera de medicina, a raíz de la Guerra Mundial Klein perdió su herencia y se puso a trabajar en circos, teatros y en la calle, mostrando sus excepcionales cualidades. Tras escarceos en empleos estables como calculista, siempre terminaba en la calle o en un circo cualquiera. En cierto momento fue descubierto por el director del CERN. Klein le pidió trabajo en el organismo, y el director accedió a tomarlo a prueba. El señor Klein resultó muy útil en el Centro de Proceso de Datos, aprendió programación, y era capaz de evaluar pedazos completos de programa. Terminaremos por decir que el señor Klein aparece en el *Libro de Records Guinness* varias veces por realizar cálculos rapidísimos, mejorándose a sí mismo varias veces.

cado de este tipo de máquinas. (Seamos leales e indiquemos que Ramón Vereá, pese a ser español, estaba afincado en Nueva York.)

El siguiente avance sustancial en la técnica se debió a otro americano, Herman Hollelith. Sin embargo, hemos considerado más



Calculadora de Bourroughs.

oportuno hablar de él en el próximo número, ya que apenas si tendríamos espacio en éste. Cerremos, pues, el tema de las calculadoras, que a partir de ese momento han seguido un camino distinto al de los ordenadores, aunque siempre adoptando aquellas técnicas y sistemas que podían adaptarse al tipo de trabajo que realizaban. Desde estas antiguas calculadoras hasta las modernas pequeñísimas, que realizan multitud de operaciones complicadas, hay tanta distancia como la existente entre los rodillos para mover piedras al coche de nuestros días. Las calculadoras primeras fueron absolutamente mecánicas. Algo más tarde fueron electro-mecánicas, y finalmente han adoptado la tecnología integrada que les ha permitido tener un tamaño muy pequeño (desbancando totalmente a la regla de cálculo; tan utilizada hace veinte años, que su manejo se estudiaba a conciencia en las escuelas técnicas).

Además, su precio se ha ido reduciendo tanto que las simples calculadoras que realizaban los mismos cálculos que las primeras de Ohdner o Bollée pueden comprarse por menos de

dos mil pesetas. Las científicas siguen teniendo precios algo más altos, pero también irrisorios, ya que las más complejas, que eran programables, han sido también desbancadas por los pequeños ordenadores personales, que, aunque es cierto que precisan de un monitor o televisor, tienen tamaños bastante pequeños y realizan más tipos de tareas. Ya hemos hablado en otro momento de la caída de precios de estas máquinas. No pensamos volver sobre ello, pero sí recordar que el mejor momento de las calculadoras fue al comienzo de los años setenta, y que para el año 1975 había más de treinta y cinco millones de calculadoras funcionando sólo en los Estados Unidos. Con la aparición del ordenador doméstico, el interés del usuario ha pasado de la calculadora al ordenador. Sin embargo, hemos de preguntarnos, ¿realmente es importante para cada uno de los treinta y cinco millones de americanos que disponían de calculadora (con más funciones que las elementales) conocer la raíz cuadrada de un número cualquiera? Al menos, el ordenador hace muchas cosas más.

Lenguajes de configuración



OMO continuación de nuestro comentario sobre el proceso en paralelo para los programas en tiempo real que manejan gran cantidad de datos, vamos a ver más concretamente las características básicas que han de

tener los «lenguajes de configuración de sistemas» utilizados en el diseño y distribución de tareas para su proceso en paralelo.

Una de las funciones que deben cumplir las configuraciones es la de identificar módulos a partir de los cuales se construye el programa. Para ayudar al programador a abstraerse de los aspectos funcionales del módulo que haya de programar para la implementación del algoritmo, dentro de una configuración determinada se debe especificar las características del interfaz de cada módulo. Un interfaz define el tipo de datos y la clase de acciones que deben tomarse en un módulo, y todas las comunicaciones deben realizarse a través de este interfaz. El interfaz debe ser la única información necesaria a nivel de configuración para utilizar el módulo. Aunque el interfaz de un módulo se especifica normalmente dentro del módulo a través de ciertas listas de datos a importar y exportar, puede repetirse dentro de la configuración, para mejorar la legibilidad de ésta. Con idéntico fin de facilitar la tarea de programación, se deben poder crear varios ejemplares del mismo módulo. En este caso, cada gránulo debe tener asignado un nombre único que lo identifique dentro de la configuración.

Los programas formados por numerosos componentes (que realizan cada uno un proceso) son mucho más fáciles de compren-

der y mantener si los subgrupos de gránulos pueden ser considerados globalmente como gránulos individuales. Los módulos se pueden combinar hábilmente anidándolos. Cada configuración anidada define un programa paralelo y, por tanto, se puede considerar como un simple gránulo, con un interfaz bien definido a nivel de la configuración. De nuevo, este interfaz es la única información lógica necesaria a nivel de la configuración global para utilizar esta estructura anidada. Esta característica también reduce simplemente a unas especificaciones de configuración los cambios que hay que realizar en los casos en que los gránulos componentes se intercalen, fusionen o dividan; para mejorar las características del programa paralelo.

Otra característica fundamental muy útil que dirige al programador hacia la construcción modular de los programas paralelos es que los gránulos se puedan compilar y probar independientemente. Es importante que hagamos notar que no es lo mismo compilar de forma independiente que de forma separada. Esta última opción obliga a llevar un orden de compilación implícito entre los módulos, ya que la compilación de un módulo puede necesitar información de otros módulos. El primer tipo permite la compilación y comprobación de los módulos en cualquier orden, y las comprobaciones entre las fronteras de los módulos se establecen durante la fase de configuración.

Como ya hemos indicado que la única información necesaria para combinar módulos es su interfaz, no existe razón alguna por la que todos los módulos individuales deban estar escritos en el mismo lenguaje de programación. Así, los módulos componentes de muchos programas en tiempo real están escritos en varios lenguajes, cada uno en el lenguaje de programación más útil (en el que pueden

resultar más funcionales). Por ejemplo, los gránulos de planificación podrían estar escritos en LISP, mientras que los gránulos de sensores e interfaces de actuación externa pueden estar escritos en lenguajes procedurales, como, por ejemplo, C, o PASCAL. El configurador debe proveer un interfaz uniforme para varios lenguajes de programación, de forma que los programadores tengan libertad absoluta para seleccionar el lenguaje de programación que pueden utilizar para cada uno de los módulos.

Veamos cómo puede estructurarse un programa paralelo formado por numerosos gránulos. Para ejecutar un programa paralelo, naturalmente se deberán crear los gránulos que conforman dicho programa. Estos gránulos se pueden crear de dos formas distintas: de forma estática, y así su número, tipo e identificación se conocerán y fijarán desde el mismo momento de su configuración. O bien los gránulos se pueden crear también de forma dinámica, según vayan produciéndose las necesidades durante la ejecución. La creación de gránulos dinámicos permite gran flexibilidad en la construcción de los programas paralelos, especialmente cuando el número de gránulos requeridos depende de las entradas al programa. El inconveniente en este caso es que la estructura general del programa se hace difícil de comprender, ya que las instrucciones de creación de cada gránulo quedan ocultas dentro del código del módulo. También obstaculiza el análisis del flujo estático necesario para la evaluación del rendimiento y detección de los estancamientos o puntos muertos, ya que el número de gránulos activos no se puede determinar estadísticamente. Más aún, el uso indiscriminado de estas características de creación dinámica pueden hacer que la creación de un gránulo se posponga de forma incontrolada, debido a tener que esperar para obtener un recurso de hardware. Por todas estas razones es imposible predecir el comportamiento de un programa en tiempo real. En los casos en los que sea necesario predecir el rendimiento bajo las «peores condiciones de funcionamiento, será necesario declarar todos los gránulos desde el momento de la configuración».

Estudiemos ahora topologías de una red de gránulos.

Una configuración se puede considerar como una especie de «red local» formada por gránulos. Los nodos de esta red serán los gránulos componentes, y los arcos entre todos re-

presentan redes de comunicación. Las relaciones entre los módulos desde el punto de vista de comunicaciones, de sincronización, etc., se denominan «topología de la red de módulos». Dentro de este contexto, en las especificaciones de dicha topología, deben considerarse dos aspectos importantes para las comunicaciones, el mecanismo de asignación de nombre, llamada y la posibilidad de cambiar dicha topología durante la ejecución.

Los distintos módulos deben ser capaces de llamarse (durante las comunicaciones a través de los canales) mediante nombres. Dichos nombres pueden ser directos o basados en nombres globales o locales. Los nombres directos pueden ser de «una vía» o de «dos vías». Los nombres de una vía permiten que el emisor pueda nombrar (llamar) al receptor, o viceversa. Por el contrario, si es de dos vías, tanto receptor como emisor se podrán nombrar uno a otro. La nominación llamada directa elimina la independencia de interconexión entre gránulos de programación individuales. En particular, no es útil para desarrollos de programas muy grandes desde el final hacia arriba, cuando los módulos no se conocen hasta el momento de la configuración. La independencia en la interconexión es un factor importante para desarrollar módulos de programas que sean reutilizables. La utilización de nombres globales significa que los gránulos se comunican utilizando un único nombre que es válido (a nivel global) para todos los gránulos, ya sean éstos «buzones» (módulos de almacenamiento de la información) compartidos, puertas de comunicación o procedimientos remotos. Los nombres globales reducen la independencia en las interconexiones, ya que éstos nombres únicos están compartidos entre los gránulos componentes. Por ejemplo, no es posible crear varios ejemplares del mismo módulo de forma que cada uno de ellos se comunique con gránulos distintos.

Los nombres locales permiten libertad en las interconexiones separando las especificaciones sobre la topología de la red, de la programación de los gránulos componentes. Con este sistema, cada gránulo utiliza para su comunicación puertas definidas localmente, y de este modo ya es posible utilizar los mismos nombres de puerta dentro de gránulos diferentes. Para enviar o recibir un mensaje a través de una puerta local, ésta debe estar conectada a puertas locales de otros gránulos. Las uniones entre puertas se pueden establecer en el momento de la configuración, o en

el de la ejecución. Si se hace en el momento de la configuración, las uniones entre puertas estarán definidas dentro de la configuración, y la forma en que estén conectadas las puertas no afectará a los gránulos individuales. Muchos lenguajes configuradores, como (por ejemplo, el PCL, NESTLA, CONIC o DICON) soportan este método para establecer las uniones. Si se hace, por el contrario, durante el momento de la ejecución, las capacidades para enviar mensajes entre gránulos se asignan a variables locales durante la ejecución. Para el envío de mensajes sólo se pueden utilizar variables «de comunicación» y con características adecuadas. Este procedimiento se ha visto que es útil en la construcción de sistemas operativos. El sistema MODEL soporta ambos métodos.

Volvamos a las dos posibles estructuraciones de las topologías de que hablamos anteriormente.

La topología de la red se puede establecer de forma estática o dinámica. Si se hace por el segundo sistema, significa que las interconexiones entre gránulos se pueden cambiar durante la ejecución del programa, es decir, un gránulo podrá comunicarse con distintos gránulos en momentos diferentes, utilizando los mismos comandos de comunicación. La topología estática no puede alterarse una vez establecida, de forma que el destino de cada comando de comunicación ha sido ya fijado para toda la ejecución completa. Es importante hacer notar que la creación de gránulos dinámicos o estáticos no da en absoluto como resultado el establecimiento de redes topológicas estáticas o dinámicas (o al contrario). Por ejemplo, se pueden crear nuevos gránulos en el momento de la ejecución, pero fijando sus interconexiones en el momento de compilar (esto puede hacerse en lenguajes como PCL o Ada). Del mismo modo, también se pueden crear todos los gránulos en el momento de la compilación, y cambiar las interconexiones, si las posibilidades de comunicación se pueden modificar mediante mensajes, como en Demos, MODEL, o NIL.

Existen distintos métodos de soportar topologías dinámicas. El más común es permitir que los comandos de comunicación contengan una variable que denota su destino, y

cuyo valor puede cambiarse en el momento de la ejecución. El inconveniente que tiene es que las actividades que controlan la topología de la red están entremezcladas con las de los gránulos individuales. Esa característica hace que los programas sean difíciles de comprender, ya que las actividades de creación de los gránulos no están disponibles para su examinador por alguien que intente comprender el programa. Existen, sin embargo, métodos que soportan tipologías dinámicas a nivel de la configuración. Por ejemplo, en CSL y en NESTLA la configuración incluye una secuencia de instrucciones ejecutables que controlan la topología de la red, según va avanzando la ejecución del programa. Con este sistema, la topología de gránulos se va redefiniendo dinámicamente, dependiendo de cuál sea la parte de la configuración que se está ejecutando en ese momento. En CONIC, la configuración (incluyendo las especificaciones de topología de los gránulos) se pueden sustituir dinámicamente por la nueva configuración. La utilidad consiste principalmente en que la topología de los gránulos se pueden ir ajustando a entornos que cambian dinámicamente, incluso en el caso de que los cambios no se conozcan por anticipado (cuando se construyan inicialmente las configuraciones).

Las topologías estáticas tienen dos limitaciones fundamentales. En primer lugar, impiden que un gránulo se comunique con aquellos gránulos que no se conocen en el momento de la compilación, limitando de este modo la posibilidad de funcionamiento del programa en un entorno que cambia dinámicamente. En segundo lugar, si en un momento dado un gránulo necesita comunicarse con otro, la previsión de comunicación debe ir incluida dentro de la propia topología. La capacidad de cambiar la topología de la red dinámicamente es muy importante en algunas aplicaciones, como, por ejemplo, en ficheros de envíos por correo, que están vigentes por mucho tiempo, pero que suelen cambiar cada cierto tiempo. Sin embargo, existen muchas otras aplicaciones, que no necesitan estas particularidades. La topología dinámica realiza análisis del flujo estático, para detectar posibles puntos muertos, y también análisis de rendimientos, prácticamente inviábiles en programas que no sean absolutamente triviales.

GLOSARIO DE TERMINOS UTILIZADOS EN EL PROCESO PARALELO

Bloqueante. (Ver **Primitivas.**)

Configurador. Lenguaje en el que se diseña el conjunto del proceso, y que prepara las diferentes tareas, define la separación de módulos los interfaces, etc. Existen numerosos lenguajes utilizables con esta finalidad. Entre ellos los más usuales son PCL, NESTLA, CONIC, DICON, MODEL...

Emisión (de mensajes). (Ver **Mensajes.**)

Interfaz. Dispositivo o programa que sirve de enlace entre dos procesos o equipos. En el proceso paralelo son básicos los «interfaces» que realizan la comunicación entre los diferentes procesos o gránulos que se están procesando en paralelo. Una de las claves del éxito de la programación de procesos en paralelo es el adecuado diseño de estos interfaces.

Mensajes. Bloques de información enviados de un gránulo a otro para comunicación y transmisión de resultados parciales, durante el proceso general.

No bloqueante. (Ver **Primitivas.**)

Nombre. Identificador de los diferentes módulos o gránulos utilizados por los restantes gránulos o módulos en el envío de mensajes. Los nombres pueden ser directos o in-

directos; y los primeros «de una vía» o «de dos vías».

Primitivas. Se llaman así a los comandos utilizados en el proceso paralelo para la comunicación de unos gránulos con otros. Las primitivas pueden ser «bloqueantes» o «no bloqueantes», dependiendo de que paralizen la ejecución del gránulo receptor o no, cuando llegan a su destino. Pueden ser también explícitas o implícitas, dependiendo del modo en que se envían. Por último, en función de la acción que realizan (enviar mensajes, recibirlos, devolver respuestas, etcétera) se les denomina emisoras o receptoras.

Remota. Llamada. Procedimiento de comunicación entre gránulos mediante el cual el módulo que envía el mensaje espera hasta recibir una respuesta explícita del receptor.

Topología. En proceso paralelo, es la distribución e interrelación de los gránulos dentro del sistema total diseñado. Los elementos que definen la topología del sistema son la estructura de estos gránulos y las comunicaciones que entre ellos se establecen. La topología puede ser estática o dinámica, dependiendo de que las interrelaciones y comunicaciones entre los módulos se predefinan en la etapa de diseño y compilación, o se vayan estableciendo por parte del propio sistema, a medida que se va ejecutando

VOCABULARIO DE INFORMATICA

Jerarquía. Orden de precedencia. Véase cada uno de los tipos de jerarquía.

Jerarquía de datos. Estructuración de datos en conjuntos, subconjuntos, etc., de forma que su organización básica sea jerárquica. Esta estructuración puede encontrarse en las especificaciones lógicas (formales), en las físicas (estructura real de almacenamiento de los datos), o en ambas.

Jerarquía de funciones. Secuencia de conjuntos de funciones que tienen la propiedad de que cada uno es igual o está contenido en el siguiente, y así sucesivamente.

Las jerarquías de las funciones recursivas primitivas pueden definirse haciendo que una función cualquiera represente las funciones que pueden computarse por programas que contienen bucles anidados. Si la función es F_n , podrá tener hasta n bucles anidados. Por tanto, $F_n \leq F_{n+1}$ para n mayor que 0. La unión de todos estos conjuntos comprende todas las funciones recursivas primitivas, y únicamente esas funciones. (Véase jerarquía de Gregorczyk.) Bertrand Russell inventó la teoría de los tipos que establece que las funciones de un determinado nivel sólo pueden definirse en términos de niveles inferiores.

Jerárquica, estructura... de la memoria.

Sistema de organización de la memoria en diferentes niveles. Se considera la memoria (las áreas de almacenamiento de datos del ordenador) organizadas de un modo jerárquico, de tal forma que la memoria más rápida y/o accesible (registros) sea la primera en ser accedida para, posteriormente y subiendo en los diferentes niveles con tiempos de respuesta mayores, pero con coste

por unidad de almacenamiento menores. Se suelen considerar niveles sucesivos en la organización jerárquica de la memoria: los registros de procesos, los grupos de palabras de las memorias intermedias, la memoria principal, las páginas de memoria almacenadas en los dispositivos de almacenamiento externo auxiliar y los ficheros completos en estos dispositivos.

Jerárquico, direccionamiento. Método para identificar la posición de cada uno de los componentes de una red que utiliza un sistema jerárquico, es decir, las direcciones se agrupan para reflejar las relaciones entre las distintas entidades.

Jerárquico, sistema de bases de datos.

Sistema de gestión de bases de datos en el que la organización de la información es jerárquica, es decir, cada registro pertenece a otro.

Jerárquico, sistema de comunicación.

Estructura de comunicaciones formada por varios niveles. Cada nivel superior cubre un área más general de operación que el nivel inmediato inferior.

Jitter. Término inglés para indicar perturbaciones, o pequeñas variaciones rápidas de la forma de la onda debidas a multitud de causas, entre ellas fluctuaciones de la tensión de alimentación, inestabilidad del sistema de control, etc.

Job (trabajo, tarea). Término inglés que indica un grupo de tareas que debe realizar el ordenador y que forman una unidad. Por extensión, puede significar todos los programas, módulos cargables, archivos e instrucciones que necesita el sistema operativo.

VOCABULARIO DE INFORMATICA

Job control, instrucción de. Instrucción perteneciente a un job y que se utiliza para identificar el job o para describir al sistema operativo sus requerimientos funcionales.

Job control, flujo del. Término inglés para designar el control de flujo de trabajos, es decir, el orden de ejecución de las distintas tareas por parte del ordenador. Este orden tiene como objetivo la utilización óptima de los recursos.

Job management. Término inglés para indicar la gestión de las distintas tareas que debe realizar el ordenador. Conjunto de procedimientos y normas que constituyen la base para la ejecución de los trabajos que debe realizar el ordenador.

Job processing control. Términos ingleses usados para indicar una parte del programa de control que se ocupa de iniciar las distintas operaciones, asignar recursos y pasar a la siguiente tarea.

Job step. Unidad de trabajo, considerado desde el punto de vista de usuario, no de máquina. Cada una de las tareas sucesivas en que se puede dividir un trabajo, y que el «flujo de job control» maneja y gobierna.

Joystick. Dispositivo periférico muy versátil parecido a las palancas de manejo de los aviones. Permite introducir al ordenador las coordenadas de posición de forma continua, rápida y cómoda.

Jump. Término inglés que significa salto. Las instrucciones salto provocan el paso de una instrucción a otra especificada, que no tiene por qué ser consecutiva. Desviación de la secuencia normal de ejecución.

Justificar. Término derivado del inglés «justify», que significa alinear unos caracteres dentro de unos márgenes dados. Puede tratarse de un único margen, por la derecha o por la izquierda, quedando el otro extremo del campo sin rellenar completamente, en el caso de que el campo a rellenar sea mayor que los datos que se «justifican» o alinean en él. Por extensión, puede referirse a un registro, cuyo dígito más o menos signi-

ficativo se encuentre en una posición específica de dicho registro.

Karnaugh, mapas de. En lógica, diagrama de funciones y variables que solapa rectángulos, etc., de forma que cada intersección de las figuras superpuestas representa una única combinación de variables lógicas, y de tal forma que quedan evidentes las intersecciones de todas las posibles combinaciones. Se utiliza para la representación de las funciones lógicas (especialmente con vistas a su simplificación).

Kbit. Unidad de medida del tamaño de la memoria (kbit) equivalente a 1024 bits (dos elevado a diez).

Kbyte. Indicador de la medida de la memoria equivalente a 1024 bytes.

Key. Término inglés que puede referirse tanto a una tecla del teclado de su ordenador como a una clave, o leyenda de un dibujo, o a otra clave cualquiera de acceso o control (palabra clave, etc.).

Keyboard. (Véase **Teclado**.)

Key in. Término inglés que designa una operación por la que se introducen los datos al ordenador de forma manual a través del teclado.

Keypad. Término inglés que suele indicar un teclado numérico.

Keyword. Término inglés que significa palabra clave. Véase, por tanto, dicha referencia.

Keyword, parameter. Parámetro de palabra clave. Parámetro que se identifica por un nombre o palabra clave en lugar de hacerlo por su posición en la lista de parámetros.

Kilo. Prefijo que multiplica el valor que le sigue por 1.000. En informática, el término se multiplica por 1.024, es decir, 2 elevado a 10.

Knowledge engineering. Término inglés que significa ingeniería del conocimiento. (Véase **Sistema experto**.)

